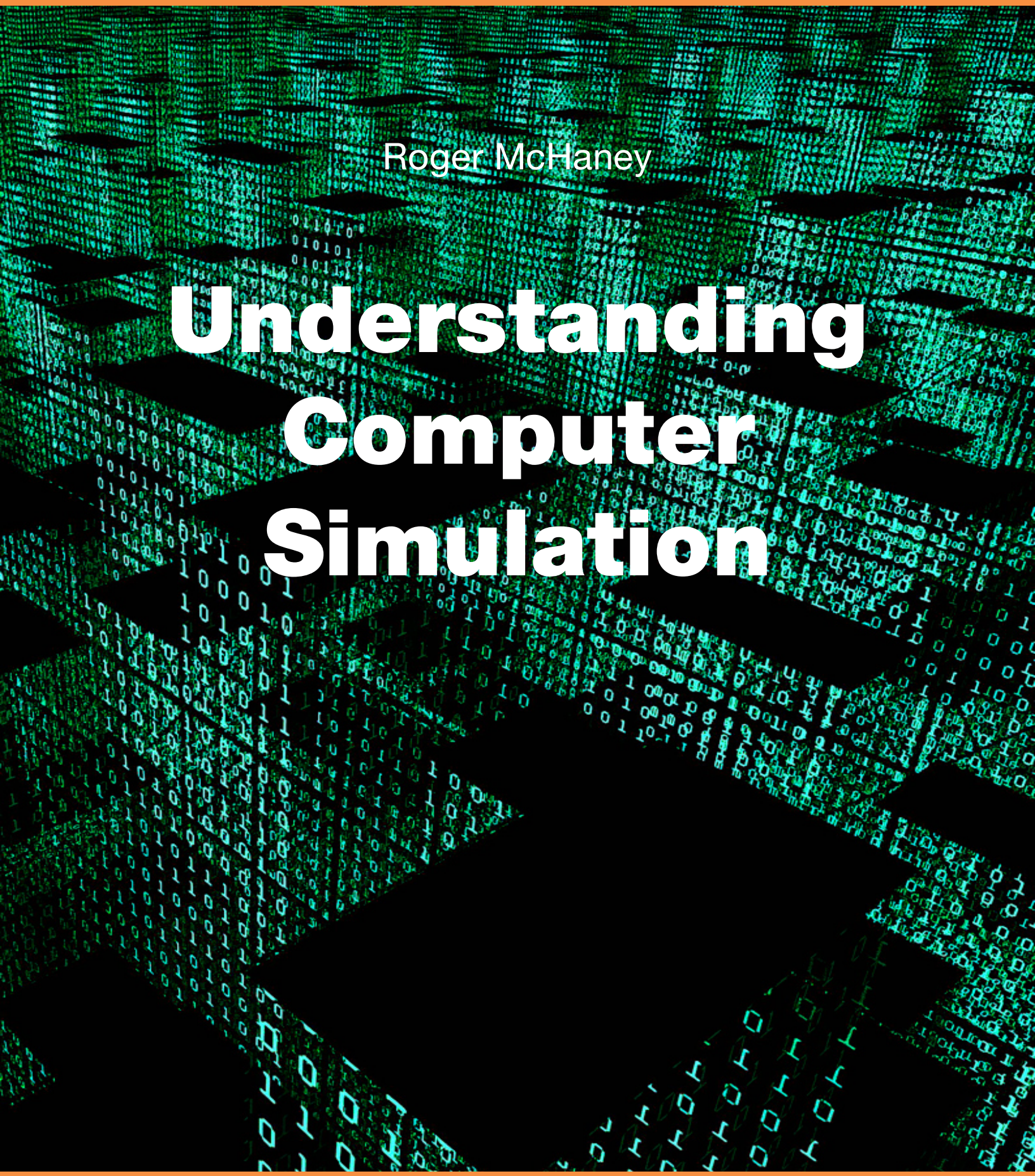


Roger McHaney

Understanding Computer Simulation

bookboon
LEARNING

Roger McHaney

Understanding Computer Simulation

Understanding Computer Simulation
1st edition

ISBN 978-87-7681-505-9

Contents

1 Introduction to Computer Simulation

Computer simulation is used to reduce the risk associated with creating new systems or with making changes to existing ones. More than ever, modern organizations want assurance that investments will produce the expected results. For instance, an assembly line may be required to produce a particular number of autos during an eight hour shift. Complex, interacting factors influence operation and so powerful tools are needed to develop an accurate analysis. Over the past few decades, computer simulation software, together with statistical analysis techniques have evolved to give decision makers tools equal to the task. As the world grows more technical and the need for precision becomes more important, the margin for error will continue to shrink. Business, industry, and governments cannot afford to make educated guesses during systems development. For that reason, computer simulation is more important than ever.

1.1 Simulation Defined

Simulation uses a model to develop conclusions providing insight on the behavior of real-world elements being studied. Computer simulation uses the same concept but requires the model be created through computer programming. While this field has grown and flourished with availability of powerful software and hardware, its origins in the desire to forecast future behaviors, run quite deep.

Men and women have attempted to foretell the future since ancient times. Kings employed wizards and soothsayers. Various religions used prophets. Seers such as French apothecary Michel de Nostredame (better known as Nostradamus) became famous with their visions of the future. Others attempted to make predictions based on birth dates and the stars. Crystal balls, bones, and tarot cards were all used as tools to probe the future.

Although this book does not advocate those methods, in the same way modern chemists bear a relationship to the ancient alchemist, the modern simulation practitioner has a relationship with the ancient prophet. Of course, methodologies used by modern simulation analysts bear virtually no similarity with the prediction methods used in ancient times. However, there are common elements. For instance, each sought to remove the risk of a future event or behavior and reduce uncertainty. The prophet tried to accomplish this with the magic available at the time. Today, the simulation analyst uses the modern magic of mathematical principles, experimentation, computer science and statistics.

1.1.1 Computer Simulation's Basic Nature

Computer simulation can be classified as a branch applied mathematics. The use of computer simulation increased due to availability of computing power and improvements in programming languages. Added to this are inherent difficulties or even impossibilities to accurately describe complex real world systems using analytical or purely mathematical models. For these reasons, a tool that can represent these complexities accurately is required. Computer simulation can be broadly defined as:

"Using a computer to imitate the operations of a real world process or facility according to appropriately developed assumptions taking the form of logical, statistical, or mathematical relationships which are developed and shaped into a model."

The result can be manipulated by varying a set of input parameters to help an analyst understand the underlying system's dynamics. The model typically is evaluated numerically over a simulated period of time and data is gathered to estimate real world system characteristics. Generally, the collected data is interpreted with statistics like any experiment.

1.1.2 Computer Simulation Uses

Computer simulation can be an expensive, time consuming, and complicated problem solving technique. Therefore, certain circumstances warrant its use. Situations well suited to its application include the following (Table 1.1):

General Situation	Examples
Real system does not yet exist and building a prototype is cost prohibitive, time-consuming or hazardous.	Aircraft, Production System, Nuclear Reactor
System is impossible to build.	National Economy, Biological System
Real system exists but experimentation is too expensive, hazardous or disruptive to conduct.	Proposed Changes to a Materials Handling System, Military Unit, Transportation System, Airport Baggage Handling System
Forecasting is required to analyze long time periods in a compressed format.	Population Growth, Forest Fire Spread, Urbanization Studies, Pandemic Flu Spread
Mathematical modeling has no practical analytical or numeric solution.	Stochastic Problems, Nonlinear Differential Equations

Table 1.1 Situations Warranting Computer Simulations

1.1.3 General Benefits of Computer Simulation

Using computer simulation for analysis has many advantages over other decision making techniques. Among these advantages are:

1. Allows Experimentation without Disruptions to Existing Systems – In systems that already exist, testing new ideas may be difficult, costly, or impossible. Simulation allows a model to be developed and compared to the system to ensure it accurately reflects current operation. Any desired modifications can be made to the model first, the impact on the system examined, and then a decision to implement the changes in the real world system can be made.

Example: Changing the Line

An automotive assembly line may run 24 hours a day, seven days a week. Shutting the line down, putting in a temporary modification (which may or may not speed up a process), and resuming production would be very expensive.

Example: Adding Equipment

Unless the machinery was already purchased, the process of adding it to the line and trying it firsthand would be impossible.

2. Concept can be Tested Prior to Installation – A computer simulation will allow concepts to be tested prior to the installation of new systems. This testing may reveal unforeseen design flaws and give designers a tool for improvement. If the same flaws were discovered after installation, changes to the system might end up being very costly or even impossible to implement.

Example: Purchase of an Automatic Storage and Retrieval system (ASRS)

Engineers in a medium sized company decide to replace an existing warehouse with an ASRS. After stretching their tight budget to its outer limits, the equipment was procured and installed. Several months of usage revealed the new system was unable to keep up with the demands for pallets being entering and removed from the racks. Their assembly line process began to bog down because material wasn't arriving from storage in a timely fashion. The budget was gone and upper management told manufacturing to live with their decision. The entire situation could have been avoided by simulating the ASRS system prior to purchase.

3. Detection of Unforeseen Problems or Bugs – When a system is simulated prior to installation and found to work in concept, the model is often refined to include finer details. The detailed simulation may reveal unforeseen problems or bugs that may exist in the system's design. By discovering these problems prior to installation, debug time and rework costs can be avoided. In addition, improvements to system operation may be discovered.

Example: Traffic light simulation

Analysts were running data through the model of a new traffic light to be installed near a busy industrial park. Traffic seemed to flow smoothly most of the time but one exception existed. On average, daily traffic flows were heaviest from east to west. However, the largest factory had a shift change at 3:30 PM and that caused traffic to run heavier in the north and south directions. By detecting this fact, analysts were able to produce a recommendation that would lengthen the north and south green light times between 3:00 and 4:00 PM Traffic would run smoother and congestion would be eased.

4. Gain in Knowledge on System – A primary benefit of the simulation process is an increase in overall system knowledge. At the start of a simulation project, especially in modeling of complex systems, knowledge is often dispersed among many different people. Each individual is an expert in his or her particular area. In order to develop a working simulation, all this information needs to be gathered together then woven into a complete picture. This process of bringing all the pieces together ends up being of great value by providing those involved with an education on the system. The simulation analyst ends up being a sleuth seeking out information from various sources to produce a finished picture. In the situation where simulations are conducted on a regular basis, channels for the information gathering process need to be established. This will speed up the process considerably and allow data to flow from individual experts to simulation.
5. Speed in Analysis – After a model has been developed, it is possible to run the simulated system at speeds much greater than would be attainable in the real world. With many simulation languages, multiple experiments can be set up and run, adding to the time savings. A finished model can take anywhere from fractions of seconds to hours of run time to produce results. But these results can represent minutes, hours, days or even years of system time. Many manufacturing plants keep a simulation of their assembly processes on hand. Each morning before starting production, engineers enter anticipated system inputs to their model and predict how long the daily operations will take.
6. Forces System Definition – In order to produce a valid, working model of a system, it is important all aspects of that system be known. If incorrect or incomplete definitions exist, the model will be inaccurate and should not be used as an analysis tool. Therefore, development of a simulation ends up forcing analysts to fully define all parameters pertinent to its operation. If certain facts cannot be determined with complete certainty, a safe guess should be calculated. When the model is run, if this guess is found to be a trouble spot causing a system bottleneck, then a more careful investigation and possibly additional research time is indicated as being necessary.
7. Enhances Creativity – Having a simulation on hand can enhance creativity in the design of a system. For instance, an engineer may conceive two possible solutions for a particular problem on the factory floor. One solution is guaranteed to work but is more expensive. The second solution involves a new technology which is less expensive, but somewhat risky. Without any means of analyzing the two possible courses of action, the more conservative one will be chosen. If a model of the system is available, both potential solutions could be tried and compared. If the less expensive one was found to perform as desired, then the creativity of the engineer could be exercised without the risk of failure. Additionally, other more radical ideas could be tested at the model level with little more ventured than the time of the analyst.

All these simulation advantages have a common thread: the reduction of risk. Simulation is a risk reduction method. Uncertainty is reduced and replaced with certainty about the expected operation of a new system or about the effects of changes to an existing system.

1.1.4 General Limitations of Computer Simulation

Simulation is not a perfect cure-all that works in every case to remove every risk from uncertain decisions. It has limitations and disadvantages. Some of these follow:

Expensive – The creation of a computer model often can be an expensive method of analysis. Although lower priced simulation packages are available, most large scale modeling efforts represent a major investment in training, software, hardware, analysis and development time.

Time Consuming – Modeling does not always produce quick answers to questions. In most cases, data collection, model development, analysis, and report generation will require considerable amounts of time. The simulation process can be sped up through two methods: reduction of detail (scaling) and by using generic code libraries. By reducing the level of detail, general concept questions can be answered much faster. However caution should be exercised when using this approach. Model accuracy can also be affected by eliminating key details. In situations where many similar simulations will be run, a generic simulation, or code library can be created. This reusable resource will prevent reinventing the wheel for each simulation project. This is the main idea behind the use of simulators.

Yields Only Approximate Answers – Discrete event simulation relies on the use of random number generators to provide model input. Since the input has a random element, some uncertainty is also associated with the output. In order to produce meaningful results, statistics must be used as a tool for interpreting output. All outputs are estimates of the true behavior of the system. It is important to recognize this fact and treat simulation results as close approximations and use statistical testing to draw conclusions.

Difficult to Validate – Validation is the process of making sure the computer model accurately represents the system being studied. When the system does not yet exist, this can become a formidable task. The opinions of experts and intuition must be relied on as a means of insuring the model runs in the same way the system will. Whenever judgment and opinion are being used, the possibly of error exists. It often is better to take a conservative point of view and make sure the system's predicted operation is no better than expectations for actual operation. Additionally, the human element in computer simulation can be difficult to accurately capture and model.

Accepted as Gospel – Another problem that can occur over the course of a simulation study is the tendency of users to accept output as gospel. Simulation is a tool used by humans and is subject to any error that a person can make. Output reports should always be subject to rigorous review by the data user. Not only should statistical testing be employed, but common sense should be used as a mechanism for acceptance. Many times problems can be detected by thinking through the process being simulated and the formulation of opinions on what should happen. If output data doesn't appear to be in line with expectations it should be investigated more closely. In this way, a problem may come to light.

1.2 Different Types of Simulation

Computer simulation takes a variety of forms and the term has acquired various meanings in different fields. A related term, often used interchangeably with computer simulation is computer modeling. Generally speaking, computer simulation is a broad term that includes the practice of generating inputs from simulated users and then passing these values into actual computer software (this practice is sometimes called emulation). For example, flight simulators can both simulate flight operations as well as control actual flight software. Computer modeling narrows the scope of application and implies all system aspects are represented using a computer. While this book uses the broader term computer simulation, most of its context relates to computer modeling.

Computer simulations have been used successfully in many fields including engineering, production management, business, the sciences, technology, architecture, entertainment, government, military and logistics/transportation. Several types of computer simulation are commonly studied for use in engineering and business environments. These include continuous, Monte Carlo, discrete event, and agent-based modeling. Although many authorities consider discrete event to be a form of Monte Carlo simulation, they will be evaluated separately since, in practice, each is used uniquely and for different applications.

1.2.1 Continuous Simulation

Continuous simulation is concerned with modeling a set of equations, representing a system, over time. This system may consist of algebraic systems, game theoretic models, statistical models or differential equations set up in such a way as to change continuously to represent the ebb and flow of parameters associated with the system state. An example of a continuous simulation is the model of a four wheel drive suspension system in which the dynamics of running over different terrains could be examined. Continuous simulations are often used in conjunction with CAD (Computer Aided Drafting) systems or within mathematical modeling software packages.

Another example of a continuous simulation is a model of competition between two populations. Biological models of this type are known as predator-prey models. The environment consists of two populations which interact with each other. The predators depend on the prey as a source of food. If the number of predators grows too fast, the prey available will decrease and predators will starve. If the number of predators drops, the number of prey will increase. This relationship can be analyzed with a continuous simulation using partial derivatives. The mathematics of this two species system was worked out by the noted theorist, Volterra. He demonstrated that with no outside interference, a fluctuating relationship similar to the graph shown in Figure 1.2 would result.

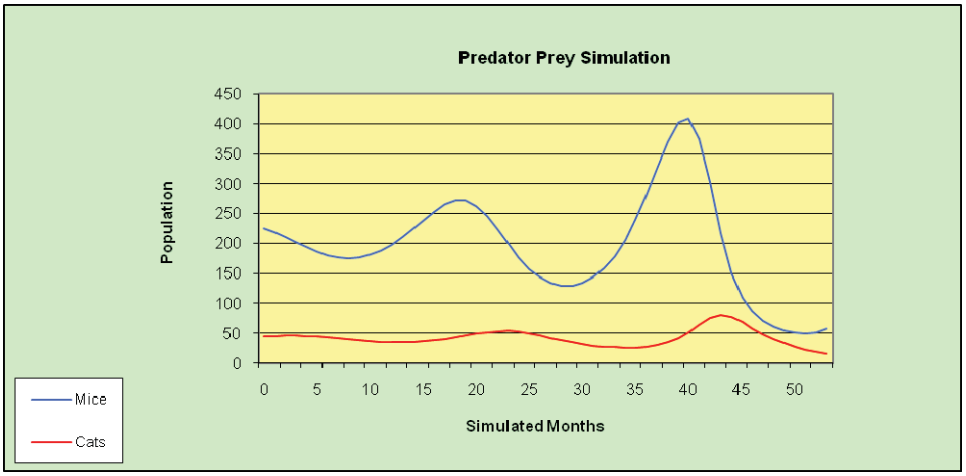


Figure 1.2 Predator Prey Simulation Model

Population growth, urban growth, hurricane predictions, weather forecasting, disease spread and fermentation models are all examples of systems that are suitable candidates for continuous simulation models. The term System Dynamics, first used by Jay W. Forrester in the 1950s, is also used to describe continuous simulation modeling. In general, system dynamics describes system behavior as interrelated, interacting feedback loops, each of which can either directly or indirectly impact another.

Continuous Simulations are commonly developed using spreadsheets, specialized mathematics software such as MATLAB or Mathematica, specialized modeling software like MATLAB's Simulink, or are developed using traditional programming languages like Visual Basic or C++.

1.2.2 Monte Carlo Simulation

The name Monte Carlo invokes thoughts of gambling, gaming and chance. John Von Neumann used the code name Monte Carlo for his experiments, based on the use of random numbers, conducted at Los Alamos during the initial development of the atomic bomb. The name became popular and is now used to represent simulations that are "a scheme employing random numbers, which is used for solving certain stochastic or deterministic problems where the passage of time plays no role" (Law and Kelton, 2000). The last part of this definition (e.g. the passage of time) distinguishes Monte Carlo from discrete event simulation. Monte Carlo generally removes time from the model, whereas discrete event simulation is based on the passage of time. The use of random number generators gives Monte Carlo simulation characteristics not common to continuous simulation.

In the Visual Basic.Net programming language, the RND() function returns a random number which can be used to simulate events which are not completely predictable but occur according to particular probabilities. For instance, consider the following paintball competition simulation written in the form of a Monte Carlo simulation.

This model predicts the numbers of Red and Blue team members to survive a match subject to the assumptions listed in Table 1.2.

1) The match is conducted by long range. Any paintballer can hit a randomly selected opponent with equal ease.
2) No two paintballers ever select the same target.
3) The match is continued until one side is completely wiped out or until 100 paintballs are fired.
4) Every paintball that hits an opposing team member removes them from the competition.
5) In the context of a Monte Carlo simulation, the entire match takes place in zero simulated time.
6) The probability of "which side shoots a paintball next" is based on the percentages of Red and Blue team members who are currently active and part of the match.
7) Initial team sizes are varied and recorded.
8) Average ending team sizes are based on 1000 matches.

Table 1.2 Monte Carlo Simulation Assumptions

Figure 1.3 provides a screen capture of the user interface for the Monte Carlo simulation of the paintball match. The source code, written in Visual Basic.Net, is shown in Figure 1.4.

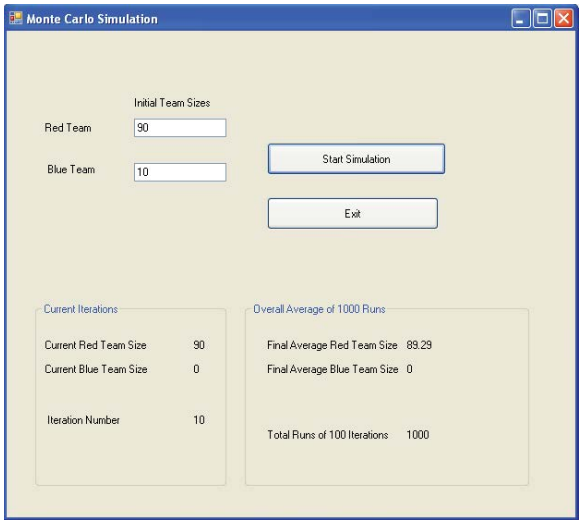


Figure 1.3 User Interface for Paintball Match Monte Carlo Simulation

```
Button1                                                    Click
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim BlueTeamSize As Decimal
        Dim RedTeamSize As Decimal
        Dim PercentPlayersLeft As Decimal
        Dim RandomDraw As Decimal
        Dim OverallAverageReds As Decimal = 0
        Dim OverallAverageBlues As Decimal = 0
        Dim Runs As Integer
        Dim NumberIterations As Integer
        Dim NumberRunsDesired As Integer = 1000

        For Runs = 1 To NumberRunsDesired
            NumberIterations = 0
            RedTeamSize = Convert.ToInt32(TextBox1.Text)
            BlueTeamSize = Convert.ToInt32(TextBox2.Text)
            Do While NumberIterations < 100 And RedTeamSize > 0 And BlueTeamSize > 0
                PercentPlayersLeft = RedTeamSize / (RedTeamSize + BlueTeamSize)
                RandomDraw = Rnd()
                If RandomDraw > PercentPlayersLeft Then
                    RedTeamSize = RedTeamSize - 1
                Else
                    BlueTeamSize = BlueTeamSize - 1
                End If
                NumberIterations = NumberIterations + 1
                Label9.Text = NumberIterations.ToString
                Label5.Text = RedTeamSize.ToString
                Label7.Text = BlueTeamSize.ToString
            Loop
            OverallAverageReds = (RedTeamSize + OverallAverageReds)
            OverallAverageBlues = (BlueTeamSize + OverallAverageBlues)
        Next
        Label12.Text = Math.Round((OverallAverageReds / Runs), 2).ToString
        Label13.Text = Math.Round((OverallAverageBlues / Runs), 2).ToString
        Label15.Text = NumberRunsDesired.ToString

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Me.Close()
    End Sub
End Class
```

Figure 1.4 Source Code for Paintball Match Monte Carlo Simulation

The model was run for 1,000 matches for each of the scenarios depicted in Table 1.3. The greater the number of runs, the closer the number of remaining players would converge to an exact solution. The value of this type of simulation is that a few minutes work on a computer gives an answer that would be difficult to evaluate in a real life situation. Another advantage inherent in this simulation is the ease with which parameters can be altered and different experiments run.

Initial Red Team Members	Initial Blue Team Members	Red Team Members Remaining	Blue Team Members Remaining
60	10	89.29	0.00
80	20	77.28	0.00
70	30	62.86	0.00
60	40	43.95	0.00
50	50	9.74	9.65
40	60	0.00	43.77
30	70	0.00	62.83
20	80	0.00	77.32
10	90	0.00	89.30

Table 1.3 Paintball Match Simulation Remaining Team Members

1.2.3 Discrete Event Computer Simulation

Discrete Event Simulation is characterized by the passage of blocks of time during which nothing happens, punctuated by events which change the state of the system. An example to illustrate this is a simple queuing system consisting of bank customers arriving at an automatic teller machine (ATM). Customers arrive, wait for service if the machine is in use, receive service and then depart. The following assumptions can be made:

1. Arriving customers wait in front of the ATM (if it is in use) in a single queue.

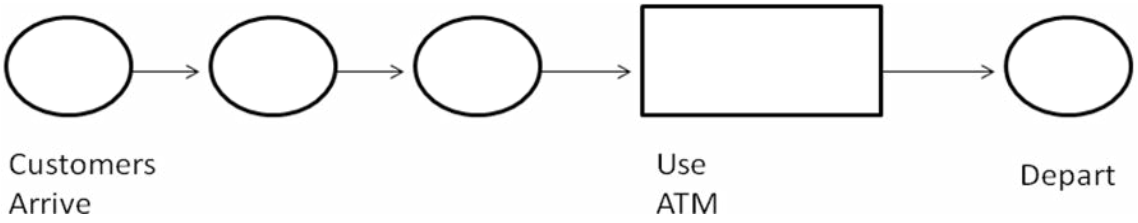


Figure 1.5 ATM Queuing System

2. The time between customer arrivals is as follows:

1 min	5% of arrivals	6 min	20% of arrivals
2 min	7% of arrivals	7 min	10% of arrivals
3 min	8% of arrivals	8 min	8% of arrivals
4 min	10% of arrivals	9 min	7% of arrivals
5 min	20% of arrivals	10 min	5% of arrivals

Figure 1.6 Arrival Times

1. The time ATM service takes is as follows:

1 min	10% of arrivals	4 min 25% of arrivals
2 min	25% of arrivals	5 min 10% of arrivals
3 min	30% of arrivals	

Figure 1.7 Service Times

2. The simulation is written in GPSS World which is a discrete event simulation language from Minuteman Software (http://www.minutemansoftware.com/).
3. The purpose of this simulation is to determine maximum customer queue length, average service wait time, and percent time the ATM machine is in use.
4. The simulation will be run for 1000 hours of simulated time.

The GPSS source code is shown in Figure 1.8. Many of the model statements may seem cryptic but in a later chapter, model construction will be discussed in detail.

```
* Sample ATM Simulation Roger McHaney
************************************************************************
* This function represents customer arrival times
************************************************************************
100    ARRIVE FUNCTION   RN2,D10
0.05,1/0.12,2/0.20,3/0.30,4/0.50,5/0.70,6/0.80,7/0.88,8/0.95,9/1.0,10
************************************************************************
* This function represents ATM use times
************************************************************************
200    SERVE  FUNCTION   RN3,D5
0.10,2/0.35,3/0.55,4/0.80,5/1.0,6
************************************************************************
* First customer is created then used to 'create' additional customers*
* according to timing specified in 'Arrive' Function                   *
************************************************************************
300        GENERATE   1,0,1,1
400 CREATE ADVANCE    FN$ARRIVE
500        SPLIT      1,ATM
600        TRANSFER   ,CREATE
************************************************************************
* Queuing and Use of ATM Machine is modeled in this segment           *
************************************************************************
700 ATM    QUEUE      WAIT
800        SEIZE      MACHINE
900        DEPART     WAIT
1000       ADVANCE    FN$SERVE
1100       RELEASE    MACHINE
************************************************************************
* Customer leaves system                                              *
************************************************************************
1120       TERMINATE
************************************************************************
* Model runs and timing are regulated                                 *
************************************************************************
1130       GENERATE   60
1140       TERMINATE  1
1150       START      1000
```

Figure 1.8 GPSS Simulation Source Code

Figure 1.9 provides a listing of the simulation's output. Even if you have never seen a simulation output before, many of the statistics are easily interpreted. For instance, the queue statistics reveal a maximum waiting line size of 5 customers. The average wait for the ATM machine was 1.381 minutes. The ATM machine was in use 76.3 percent of the time. The average length of the customer waiting line was .251 people. The machine was used 10898 times and 10899 customers entered the system. Of course, a simulation such as this can be based on actual customer arrival rates and service times observed at an ATM machine.

```
GPSS World Simulation Report - Untitled Model 1.8.1
                  Tuesday, June 09, 2009 11:12:31

           START TIME           END TIME  BLOCKS  FACILITIES  STORAGES
                0.000          60000.000    12        1          0


              NAME                      VALUE
          ARRIVE                     10000.000
          ATM                            5.000
          CREATE                         2.000
          MACHINE                    10003.000
          SERVE                      10001.000
          WAIT                       10002.000

 LABEL              LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                    1    GENERATE            1             0       0
CREATE              2    ADVANCE         10899             1       0
                    3    SPLIT           10898             0       0
                    4    TRANSFER        10898             0       0
ATM                 5    QUEUE           10898             0       0
                    6    SEIZE           10898             0       0
                    7    DEPART          10898             0       0
                    8    ADVANCE         10898             0       0
                    9    RELEASE         10898             0       0
                   10    TERMINATE       10898             0       0
                   11    GENERATE         1000             0       0
                   12    TERMINATE        1000             0       0

FACILITY         ENTRIES  UTIL.   AVE. TIME AVAIL. OWNER PEND INTER RETRY DELAY
 MACHINE          10898    0.763       4.203  1        0    0    0     0       0


QUEUE              MAX CONT. ENTRY ENTRY(0) AVE.CONT. AVE.TIME   AVE.(-0) RETRY
 WAIT                5    0  10898   6415     0.251      1.381      3.358   0
```

Figure 1.9 GPSS World Output Statistics

If the simulated ATM utilization percentage was close to 100% a decision could be made to determine if an additional machine is needed. This simulation could also be modified to represent what would happen during peak usage times. For instance, if Friday was payday at a large nearby plant, a surge of users could be expected to use the ATM after receiving their checks. Data representing this surge could be entered into the simulation model and analyzed to assess impact on machine utilization and waiting line times.

1.2.4 Agent-Based Modeling

Agent-based modeling addresses the simultaneous interactions of multiple agents to simulate, recreate, study, and predict complex phenomenon. The concept of agent-based modeling is that an overall behavior emerges through the micro-level interactions of individual agents. The primary assumption is that simple local behaviors generate complex height level behavior. Individual agents are modeled according to individual characteristics and are generally assumed to be rational, acting in their own interests which may be economic or socially derived. The model will use local heuristics and simple decision-making rules that create the larger environment.

Most agent-based models have the following elements:

1. multiple agents modeled and scaled with various levels of detail (granularity)
2. decision-making heuristics and rules
3. adaptive behaviors or learning
4. interaction rules or topology
5. environment for interaction often consisting of constrained resources

1.3 Brief History of Simulation

Generally speaking, simulation evolved from the natural human desire to remove risk from the decision making process. In ancient times, rulers often relied on prophets to foretell the outcome of a military action. In modern times, the same desire is manifested in sophisticated military models used to closely examine and statistically anticipate the outcome of particular actions and maneuvers. Methodologies have changed but the overarching goals remain constant: risk reduction and better decision making.

Early simulation efforts can be traced back to World War II when Jon Von Neumann and Stanislaw Ulam developed Monte Carlo simulation techniques to aid in their understanding and development of the atomic bomb. The modern era of simulation began in earnest during the 1950s when new concepts and methods for creating simulations were implemented with programs in available computer languages such as machine code, assembly language, or FORTRAN. Computer hardware was expensive, scarce, slow, and not always reliable. Also computer languages were not designed for simulation applications. In spite of these environmental shortcomings, the value of computer simulation became apparent. Early modeling techniques were developed and implemented on available hardware and software platforms.

Many efforts proceeded in parallel during the early days of computer simulation. Tocher was among the first to describe the application of computers to sampling experiments. These computer-based sampling experiments are now considered to be among the first true computer simulations. In the early 1960s, Geoffrey Gordon of IBM introduced a computer language called GPSS (General Purpose Simulation System) intended to manage the general overhead associated with running simulations on a computer (e.g. timing mechanisms, resource representation, entities, et cetera). This new language first was used at IBM to analyze complex systems but quickly gained widespread acceptance among various organizations and the military. In 1962, the Rand Corporation announced Harry Markowitz, Bernard Hausner, and Herbert Karr had developed the SIMSCRIPT simulation language. This software was developed as an inventory modeling tool for the United States Air Force. During this same time period, Norwegian scientists Dahl and Nygaard released the SIMULA language which, in addition to being a simulation language, was the first object oriented programming language.

The development of a simulation language industry and the realization that many parallel and similar efforts were taking place across the United States and Europe led to the establishment of workshops, support organization, and conferences intended to communicate simulation progress, reduce redundancy efforts and provide faster advances. In March, 1964 the Workshop on Simulation Languages was held at Stanford University and provided the first formal venue for developers and simulation users to exchange ideas. The need for a regular annual conference was apparent and in 1967, the first Winter Simulation Conference was held. By 1968, the Society for Computer Simulation (SCS) had became an official sponsor and gained widespread popularity as a leading organization for simulation practitioners.

Another major simulation language emerged late in the 1960s with the release of the general purpose simulation language GASP. GASP eventually evolved into the SLAM family of languages. In the 1970s, as computing power increased and hardware costs decrease, a wide variety of simulation software products entered the market. In 1977, after IBM corporation discontinued support of GPSS/V, James O. Henriksen announced a new and improved version of GPSS called GPSS/H making GPSS the first multivendor simulation language.

In the early 1980s, the advent of the personal computer led to further developments in the simulation marketplace. Two new major simulation languages were released during this time SLAM by Pritsker Corporation in 1980 and SIMAN by the Systems Modeling Corporation in 1983. Other developers targeted the largely untapped market of easy-to-use industrial simulators. During the 1980s, numerous simulation products were developed and marketed. At the same time, established simulation software companies continued to expand their product lines with animation packages, simulation development toolkits, and enhancements to existing languages. By the 1990s, the simulation market was more commercialized and segmented. More uses emerged. Simulation software fell into eight major categories with numerous offerings in each area (Table 1.4).

Simulation Software Category	Description	Example Products
General Purpose Software	Simulation languages and general software used to write simulation models.	GPSS/H, GPSS/PC, SIMAN, Simula, SLAM, SLX
Manufacturing Oriented Software	Products specifically designed for use in the analysis of manufacturing and production systems.	ProModel, AutoMod, WITNESS, ShowFlow 2.5
Planning & Scheduling Software	General purpose and specific manufacturing software tools are often used for planning and scheduling but separate software products supporting this area has emerged.	Simul8 Planner, AutoSched
Special Purpose Modeling Software / Simulators	Other specialty area simulation packages concentrate on specific areas like communications, health care, manufacturing, service industries, education, and so forth.	MedModel, a medical service environment simulation system; ServiceModel, a service industry simulation package for use with banks, schools, offices and other applications; ns-3, a network simulator
Simulation Environments	Simulation environments contain many utilities to conduct a simulation study. These capabilities include input data analysis, model entry support, scenario management, animation, and output data analysis.	Arena, GPSS/World
Animators	Animation software allows the simulation to be dynamically displayed on the screen of a computer using a graphic format.	Wolverine Software Corporation's PROOF; Arena integrates animation with underlying simulation software
Rapid Modeling Tools	Rapid modeling or rough cut modeling tools are used to develop quick models or perform feasibility studies prior to embarking on a full blown modeling effort.	Spreadsheet software is most commonly used or broadly scaled models are developed with simulation languages or simulators Manuplan and SimStarter are examples of software packages that were developed for this function but are no longer sold
Simulation Support Software	Support software includes tools to aid in the simulation process. Among these are tools used for data analysis, distribution determination, and reporting.	ExpertFit SIMSTAT 2.0

Table 1.4 Simulation Software Categories

During the 1990s, simulation product vendors focused on putting tools in the hands of end-users. Software such as AutoMod and Micro Saint gained in popularity with automatic input data collection features, programming free deployment and graphical interfaces. Additionally, web-based simulation emerged. This approach to simulation means programs are developed and executed over the Internet specifically using a web browser. The web increasingly became viewed as an environment for simulation applications.

Another growing area of simulation, agent-based modeling, began gaining popularity during the 1990s and found application in a variety of business, social, and technical areas. Agents-based models were applied to supply chain problems, consumer behavior, social interaction, workforce management, stock market analysis, pandemic group models, traffic patterns, and other areas. Agent-based models tested how changes in local behaviors impact large scale emergent behaviors. Languages such as Swarm and Repast are used in agent-based modeling.

As simulation development moved into the 2000s, the industry continued to grow both in sales and products available. Today, hundreds of simulation products are available with specialty products in numerous areas. A list of these can be found at:

http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation1main.html

1.4 Bibliography

D.T. Brunner, and J.O. Henriksen, "A General Purpose Animator," In Proceedings of the 1989 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 155–163 (1989).

S. Cox, "GPSS/PC Graphics and Animation," In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, 129–135 (1988).

O. Dahl and K. Nygaard, "Simula An ALGOL Based Simulation Language," Communications of the ACM, 9: 9, pp. 671–678 (1963).

G. Gordon, "A General Purpose Systems Simulator," In Proceedings of EJCC, Macmillian, New York, pp. 87–104 (1961).

A.M. Law and W.D. Kelton, Simulation Modeling and Analysis, 3rd Edition, McGraw-Hill Book Company (2000).

Lion Heart Publications. http://www.lionhrtpub.com/orms/orms-8-06/fragent.html

J. McLeod (Editor), Proceedings of the 1988 Conference: Pioneers & Peers, Society for Computer Simulation, San Diego, California (1988).

A.A.B. Pritsker, The GASP IV Simulation Language, John Wiley and Sons, New York (1974).

A.A.B. Pritsker, Introduction to Simulation and SLAM II, Systems Publishing Corporation, West Lafayette, Indiana (1986).

J.O. Henriksen, "An Improved Events List Algorithm," In Proceedings of the 1977 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 546–557 (1977).

T.J. Schriber, "Perspectives on Simulation Using GPSS," In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 71–84 (1988).

H.M. Markowitz, B. Hausner, and H. Karr, SIMSCRIPT A Simulation Programming Language, Prentice Hall, Englewood Cliffs, New Jersey (1963).

R.W. McHaney, Computer Simulation: A Practical Perspective, Academic Press, San Diego (1991).

R.W. McHaney, "Use cases and personas: uses in service sector simulation development," International Journal of Simulation and Process Modelling (IJSPM), Vol. 4, No. 3/4, pp 264–279 (2008).

R.M. O'Keefe, "What is Visual Interactive Simulation? (And is There a Methodology for Doing it Right?)," In Proceedings of the 1987 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 461–464 (1987).

J.J. O'Reilly and W.R. Lilegdon, "SLAM II Tutorial," In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation (San Diego, California), 85–89 (1988).

C.D. Pegden, Introduction to SIMAN, Systems Modeling Corporation, State College, Pennsylvania (1987).

C.D. Pegden, R.E. Shannon, and R.P. Sadowski, Introduction to Simulation Using SIMAN, McGraw Hill, New York (1990).

S. Robinson, "Discrete-Event Simulation: From the Pioneers to the Present, What Next?" Journal of the Operational Research Society, 56 619–629 (2005).

S. Robinson, Simulation: The Practice of Model Development and Use, Wiley, Chichester (2004).

J. Rottenbach, "Simulation Software Acts in Other Areas," Managing Automation (January), pp. 44–45 (1991).

D.A. Samuelson, and C.M. Macal, "Agent-Based Simulation Comes of Age: Software opens up many new areas of application," OR/MS Today, August (2006).

D.A. Taylor, Object Oriented information System Planning and Implementation, John Wiley and Sons, New York (1992).

M.B. Thompson, "AutoMod II: The System Builder," In Proceedings of the 1989 Winter Simulation Conference, Society for Computer Simulation, San Diego, California, pp. 235–242 (1989).

K.D. Tocher, "The Application of Automatic Computers to Sampling Experiments," Journal of the Royal Statistical Society (B:16), 39 (1954).

K.D. Tocher, The Art of Simulation, The English Universities Press Limited, London, England (1963).

2 Simulation Languages

Simulation languages are versatile, general purpose classes of simulation software that can be used to create a multitude of modeling applications. In a sense, these languages are comparable to FORTRAN, C#, Visual Basic.net or Java but also include specific features to facilitate the modeling process. Some examples of modern simulation languages are GPSS/H, GPSS/PC, SLX, and SIMSCRIPT III. Other simulation languages such as SIMAN have been integrated into broader development frameworks. In the case of SIMAN, this framework is ARENA. Simulation languages exist for discrete, continuous and agent-based modeling paradigms. The remainder of this book will focus on the discrete event family of languages.

2.1 Simulation Language Features

Specialized features usually differentiate simulation languages from general programming languages. These features are intended to free the analyst from recreating software tools and procedures used by virtually all modeling applications. Not only would the development of these features be time consuming and difficult, but without them, the consistency of a model could vary and additional debugging, validation and verification would be required. Most simulation languages provide the features shown in Table 2.1.

1) Simulation clock or a mechanism for advancing simulated time.
2) Methods to schedule the occurrence of events.
3) Tools to collect and analyze statistics concerning the usage of various resources and entities.
4) Methods for representing constrained resources and the entities using these resources.
5) Tools for reporting results.
6) Debugging and error detection facilities.
7) Random number generators and related sets of tools.
8) General frameworks for model creation.

Table 2.1 Simulation Language Features

2.1.1 Comparison to Traditional Languages

Although many models are written using simulation languages, some analysts still prefer to rely on traditional programming languages for model development. In other cases, extensions to a language are developed to add capabilities to a traditional language. For instance, Repast Simphony is a free and open source, agent-based modeling toolkit that adds features to Java in order to simplify model creation and use. This blended approach provides the advantages of both the traditional language and the simulation modeling extensions. The motivations behind using a general purpose language include:

Programmer familiarity: Developers already know the general purpose programming language. They may not have the time or inclination to learn a simulation language.

Flexibility: Programming languages inherently are flexible, giving the analyst freedom to create the model using his or her preferred methodology.

Cost: Programming language software is usually more accessible and far less expensive than specific simulation software. This may not always be true since several leading simulation languages can be downloaded for no cost. However, other leading simulation language packages can be very expensive.

Hardware Concern: General purpose software may be available on any hardware platform while some simulation languages may require special machines and memory configurations.

Lack of Analyst Knowledge: The analyst may not understand simulation languages and may lack knowledge on the advantages of using a simulation language package.

Training: Available classes in the use of traditional languages are more likely to be available than specialty simulation training.

Although traditional languages do offer some advantages, most of these are outweighed by features standard to many simulation languages. In a typical modeling application, the programmer or analyst will find the initial investment in a simulation language more than pays off. A simulation language will provide a savings in coding, debugging, analysis of results, and in making changes.

2.1.2 Simulation Languages

A variety of simulation languages exist and are used by businesses, researchers, manufacturing and service companies, and consultants. The next sections briefly discuss two common simulation languages: GPSS and SIMSCRIPT.

GPSS: General Purpose Simulation System (GPSS) was originally developed by Geoffrey Gordon of IBM and released in October of 1961. Following IBM's release of GPSS to the public domain, it became a multivendor simulation language and has been in continuous use since.

In general, GPSS enjoys widespread popularity due to its sensible world view and overall power. Its basic functions can be easily learned while powerful features make it ideal for modeling complex systems. In general, GPSS is used to simulate queuing systems that consist of customer entities interacting and completing in a system of constrained resources. The resources are structured as networks of blocks that entities (also called transactions) enter and use to perform various tasks in certain amounts of simulated time. As entities move through these networks, which have been organized to represent a real world system, statistics are collected and used to determine if the system contains bottlenecks, is over or under utilization, or exhibits other characteristics. Output data is made available for analysis at the end of a production run.

Presently, several vendors offer versions of GPSS. Included are:

> Wolverine Software which produces GPSS/H, a powerful, state-of-the-art version of GPSS engineered to allow creation of large, complex models (http://www.wolverinesoftware.com).
>
> Minuteman Software which produces a user friendly GPSS simulation environment called GPSS World that features special model development tools (http://minutemansoftware.com).
>
> ngolf Ståhl and Beliber AB which produce WebGPSS, a stream-lined version of GPSS, with a focus simulation and modeling concept education (http://www.webgpss.com).

SIMSCRIPT III: This language is a direct descendant of the original SIMSCRIPT language produced at Rand Corporation in the 1960s. SIMSCRIPT III has constructs that allow a modeler to approach a problem from either a process or an event oriented world view. SIMSCRIPT III offers unique features which add to its appeal. Among these are:

- Object-Oriented Programming
- Modularity
- SIMSCRIPT III Development Studio (SimStudio)
- Object-Oriented Simscript III graphics
- Data Base Connectivity SDBC

In general, SIMSCRIPT III is a free form language with English-like syntax. This syntax allows the code in the system to become self-documenting. Model components can be programmed clearly enough to provide an excellent representation of the organization and logic of the system being simulated. SIMSCRIPT III is maintained and distributed at http:// www.simscript.com by CACI Products Company.

2.1.3 How Simulation Languages Work

Most discrete event simulation languages model a system by updating the simulation clock to the time that the next event is scheduled to occur. Events and their scheduled times of occurrence are maintained automatically on one of two ordered lists: the current events chain or the future events chain. The current events chain keeps a list of all events that will (or may) occur at the present clock time. The future events chain is a record of all events that can occur at some point in the future. A simulation clock moves to the next event on the future events chain and changes the system state of the model based on that event's characteristics. Figure 2.1 illustrates.

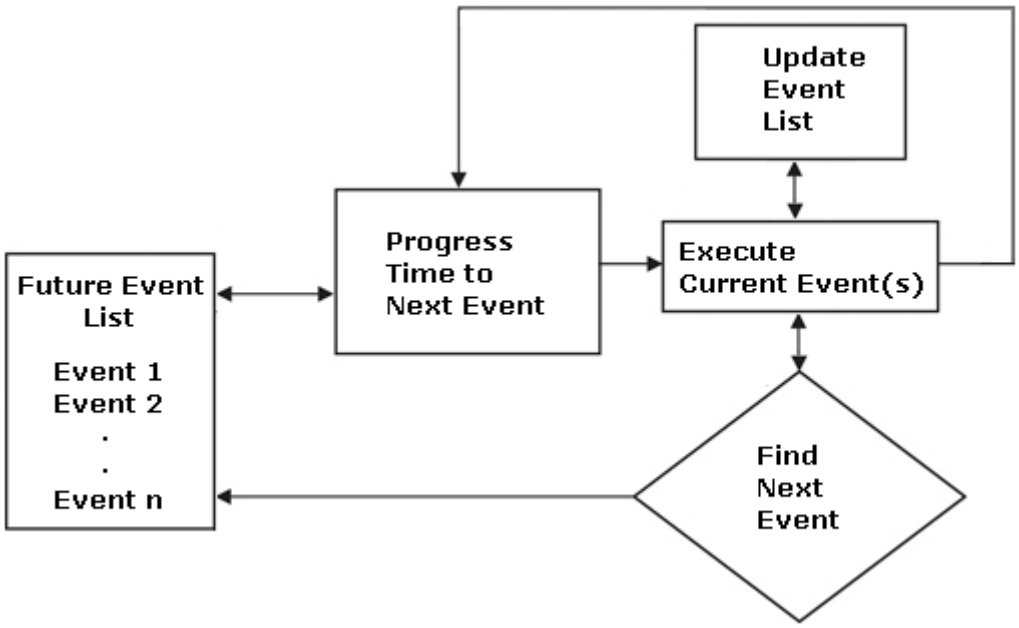


Figure 2.1 How Simulation Languages Work

Transactions are dynamic entities that traverse through the networks of blocks in a simulation. Their meaning is defined by the simulation analyst during the process of building the model. They are usually created to represent an active object such an automobile in a traffic simulation, an AGV in a manufacturing simulation, or a customer in a barber shop simulation.

Multiple transactions can exist in a model at any given time. They move through the block network as far as the current system state allows and stop either for a predetermined delay, a logical delay, or because a desired resource is currently unavailable. Parallel processes can be simulated since multiple transactions can be present in a model. Each transaction has associated with it a list of attributes which can be altered according to logic contained in the model. These parameters can be used to define unique characteristics of a particular transaction, thereby affecting its movement through the block networks.

More permanent equipment entities can be used to model constrained resources. The system state of the model at any given time is defined by the status and attributes of both permanent and temporary entities in the model.

2.2 Simulators and Integrated Simulation Environments

Simulators and integrated simulation environments have emerged to provide the analyst with additional capabilities that further automate and remove drudgery from the modeling process. Models that used to take days or weeks to develop with a simulation language can now be modeled in minutes with pre-developed representations of commonly modeled real-world items requiring no more than customization of certain parameters. New software, specialized to the application area, allows the analyst to model, execute, analyze, and animate systems in their domain (such as manufacturing, healthcare, logistics, communication, et cetera). Advanced versions of simulation software support the following:

- Environments specific to the domain area so the analyst can quickly enter the geometry, system characteristics and resource constraints for a model.
- Expert system technology uses input parameters to generate details automatically.
- Windows, help systems, and pop-up menus provide guidance and automate the modeling process.
- Facilities for comparing changes and statistical reporting tools are provided.
- Built in system specific templates make analysts more productive and eliminate programming time.
- Analysts have tools to validate, verify and test designs.
- Facilities are provided to automate and support exploration of "what if" questions being tested.
- 3-D animation graphics are automatically created from analyst inputs.
- Results can be communicated in real time using animation and statistics

2.2.1 Simulators

A simulator is defined as a user friendly software package that will develop a model for a particular application. These models generally are created by a person who is not a simulation analyst or programmer but still wishes to analyze a system. For example, the person who is an automation expert (rather than a computer programming expert) is able to model a conveyor system with pull-down menus and user friendly interface tools. MedModel (health care simulator), PRISM (police force simulator) and PX-Sim (pharmaceutical simulator) are all examples of simulator software packages. Simulators are often characterized by "buzz phrases" such as those shown in Table 2.2:

Simulator Buzz Phrases
No programming
Graphical model building interface
Model any system in just a few hours
Any manager can use it
Anyone on the shop floor can use it

Table 2.2 Simulator Buzz Phrases

2.2.2 Advantages of Simulators

Simulators offer several advantages over simulation languages. Included among these are:

Ease of Use – Simulators are designed specifically for the non-programmer. They typically have user-friendly features such as pull down command menus and special tools to simplify model construction. Many simulators are marketed with more emphasis on their user friendliness than on their underlying system modeling capability. They are meant for use by the domain expert.

Quick Model Development – Many simulators are set up to provide a fast method of constructing a model. This development speed is gained because the underlying system model has already been created and the user of the simulator is only changing parameters through a user interface. This interface may be set up to allow model construction through the piecing together of graphic icons, with a series of questions, or with series of prompts and user screens.

Base System Simulation Already Complete – At the center of all simulators is a model which has already been constructed. The user of the simulator, in theory, does not need to concern themselves with fully understanding and analyzing the system to be modeled. As long as several key attributes of the system can be identified, the simulator will do the work. This saves time, energy, and frustration on the part of the model builder.

Framework for Analysis of a Particular Type of System – Simulators have been used numerous times for the same task and therefore are constructed to provide a logical method of analyzing a particular type of system. This structure or experimental framework has evolved from repeated use and, in many cases, is superior to what a first time simulation analyst could devise. This framework is another time saving feature.

Although simulators may sound too good to be true, they offer a world of knowledge in return for little effort. When used in the proper context, they can be a useful tool. But when misused, they can provide misleading results. Some of the pitfalls of simulators are in the following list.

Oversimplification – Simulators tend to oversimplify system representation. In order to sell a simulator, a vendor must be able to apply his product to many comparable systems. This means a generic model forms the basis for the software and that it may have simplifying assumptions built in. One Automatic Guided Vehicle vendor evaluated several simulator packages only to discover the base model would not permit use of their advanced design concepts. Instead of displaying their edge over the competition, the simulator made their system like everyone else's.

Inflexible System Representation – Simulators can be somewhat inflexible. The base model has already been built and with it are some general, unchangeable, underlying assumptions. A comment was once heard from an engineer concerning the use of a simulator to aid in system design. "I'm not going to throw out my concept because the simulator can't be changed to show how it will work," he said.

Encourages Jumping the Gun – Due to the nature of simulators and the ease with which models can be created, the modeler is often tempted to skip intermediate steps in the model creation process and immediately begin performing a quick analysis. This temptation is sometimes hard to overcome and can lead to the construction of invalid models.

Visual Component May Create a False Sense of Credibility – Most simulators, especially personal computer based ones, rely heavily on the use of graphics for model creation and output displays. This visual component may give the model users a "warm fuzzy feeling" but it shouldn't be unconditionally accepted as valid. All models should be analyzed using proper validation and verification techniques.

2.2.3 Simulators vs. Simulation Languages

The following questions come to mind when trying to decide whether to use a simulator or a more robust simulation language for modeling project (see Figure 2.2).

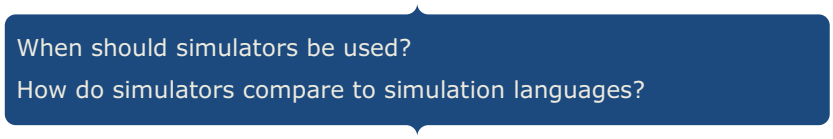


Figure 2.2 Simulator Questions

The following observations can be made:

1. Commercially available simulators should not be used for leading edge or specialty systems. In most cases, simulators are created for broad and general applications. If a modeler wants a simulation that exploits all specific attributes of his/her system to gain a competitive edge, a simulation language may provide a better fit for the project.
2. Simulators can make excellent "concept modeling tools." One of the steps in the simulation life cycle is the creation of a rough-cut or concept model. Simulators can be used to build quick models for feasibility studies.
3. Simulators can be used as estimating tools. The company that wishes to purchase a conveyor, for example, can use a simulator to provide a general estimate of what will be needed. A simulator will provide a good idea without actually getting too specific. This is particularly true when a product vendor has not yet been selected.
4. Use of a simulator does not automatically eliminate all work. Simulators reduce the time involved in model construction. The other steps in the simulation process still require time and expertise.

2.2.4 Integrated Environments

Most simulation languages and simulators are now part of broad, integrated environments intended to facilitate all aspects of the modeling process. Generally, integrated simulation environments include software tools for designing, writing, verifying, running, and analyzing models. Software environments started to appear in the simulation marketplace in the mid-1980s.

TESS (The Extended Simulation System) was one of the first integrated simulation environments to be offered. TESS originally ran in conjunction with several simulation languages and provided databases, graphics, and other integrated support features. GPSS World, produced by Minuteman software, is an example of a simulation language which has been incorporated as part of an integrated environment approach to modeling. This GPSS software package includes facilities for creation, debugging and verifying, running, analyzing, and animating of the model. Arena is another integrated modeling environment specifically configured to support manufacturing, production, and logistics.

Integrated environments allow the simulation analyst to easily move from one step in the simulation process to the next without much of the unproductive housekeeping work required by non-integrated simulation tools.

2.3 Hardware Requirements for Simulation

Computer hardware technology has advanced tremendously in the past decade. The computing power once found only in a room sized mainframe can now be carried in a small pocket sized computer. Execution speeds and memory capacity have increased while device sizes have decreased. All of these factors have helped foster an environment well suited to modeling applications.

While large scale scientific simulation work still relies on supercomputers, most manufacturing, service sector, and specialty simulators run on microcomputer platforms. Other large scale models may be run over the Internet with host computers providing required processing power. But in general, most simulation software vendors produce packages focused on Windows or Linux-based operating systems.

Microcomputers offer the ability to share simulation libraries over networks, produce high resolution graphics, and run large simulation programs. In addition, the availability of other software such as spreadsheets, statistical analysis packages, and standard programming languages gives the simulation analyst a wide variety of other tools for manipulating input data, analyzing output data, and performing other tasks in the modeling process.

2.4 Animation

Using computer graphics to dynamically display simulation entities and related activities is defined as animation. In some cases, the term simulation visualization is used for the same thing. The widespread availability of microcomputers and inexpensive, high quality graphics coupled with expectations of users to see animated systems has steadily increased the demand for a visual component in simulation.

2.4.1 Using Animation in The Proper Context

Animation is used for several reasons. Among these are:

Produces User Friendly Output – For the non-technical person, an animation will be much easier to comprehend than a statistical printout. By watching simulated entities move about on the screen, the observer can become familiar with the system's operation and with the logic embedded by the analyst.

Validation – When the simulation analyst is working with a system expert, animation provides a method of communicating information concerning the construction of the model. The expert and analyst both can visually examine the model as a means of validating its operation.

Tool for Debugging – The simulation analyst can use an animation as a debugging aid. Subtle inconsistencies may be visually detected easier than through statistics. The analyst is able to verify that the actual model matches his or her conceptual design. Animation will make errors and problems obvious, thereby eliminating the temptation to say, "It's such a rare problem that it's not worth fixing." No matter how rare or inconsequential the problem may be, if an observer can see it happen during an animation, it has to be dealt with or the face validity of the model will be destroyed.

Visual Impact/Sales Tool – One of the most popular reasons for using animation is the visual impact or sense of reality that it brings to the modeling process. It is much easier to sell a system concept when it appears to work when viewed on the screen of a computer than it is to sell a system represented by a stack of paper covered with numerical data. Most vendors of industrial equipment rely heavily on animation as a tool for use by its sales force when trying to "sell" a concept to a customer. Seeing is believing and seeing an animation is no exception.

2.4.2 Misuse of Animation

Like all tools, animation can be misused. The analyst must remember to use it to supplement a simulation study rather than drive it. The following pitfalls are common when animation is used.

Creation of Overconfidence – A model is still just a model and is only as accurate as the data and assumptions it uses. Animation tends to make model users and creators forget some of the underlying assumptions and techniques that were incorporated during the construction stage. It is easy to think that if it looks realistic on the screen then the system must be modeled properly. This is not necessarily always the case.

Unwise Time Spent on Animation – Rather than spending time building the actual model, undue time is spent trying to build an attractive animation. The animation should be used as a supplement to the simulation not as its focal point.

A Substitute for Good Statistical Work – Rather than spending time to perform a proper analysis on the results of a simulation study, the decision maker bases his or her opinions on a short period of observed animation. The animated portion of the simulation observed might not be representative of the system's true behavior. It is very important not to jump the gun and draw conclusions only on the basis of an animation.

2.4.3 Animation Attributes

Animation software may be structured in different ways depending on the application. For instance, most personal computer animators are 3-dimensional, exhibiting graphics with height, width and depth. Some animation packages offer pixel-based graphics and others, like Proof Animation from Wolverine Software, offer CAD-like 3-dimensional vector-based animation.

Another attribute of animation software is that it allows a model to run real time or in the post processor mode. The term real time is used indicate that the animation runs on the computer screen while the simulation program is executing. The disadvantages for this type of animator are a loss of speed, possible inaccurate statistical representation of the time period being modeled, and the need to run the simulation when the animation is viewed. Additionally, some animation packages allow model parameters to be changed on the fly. This can result in the corruption of output statistics.

The post-processor mode means the animation is displayed after the simulation run is complete. The simulation creates a data file that is used as an input to the animation program. A post processor offers many advantages such as the ability to fast forward, reverse, speed up or slow down a viewing. The animation can be easily transported requiring only the animator software and data file. Since the animation is run from a data file, multiple copies can be running at one time without needing additional copies of the simulation program.

2.4.4 Example Animation Software Packages

Animations are available in a wide array of simulation environments, simulators and simulation languages. Most simulation projects are accompanied with an expectation of producing an animation. Animations can be found for simulation applications ranging from flight simulators to modes coded in Second Life's Linden Script to medical simulations. To support these efforts, a variety of specialty tools exist.

On one end of the spectrum are the general graphic display objects found in programming languages such as Visual C++ or Java and the programming features to move these objects across a computer screen. On the other end of the spectrum are specialty animations developed to accompany specific models.

Many animations are created using graphics tools provided simulation software vendors. Among these are the tools found in Arena 3DPlayer and Wolverine Software's Proof Animation.

Arena 3DPlayer is a post-processor that enables a simulation analyst to create and view three dimensional animations of models created in Arena. First, a system layout is developed and when an Arena model is run, a simulation history file is created. This file feeds information into the 3DPlayer and enables the movement of dynamic, user-defined entities through the model layout. 3DPlayer provides a user-friendly interface which employs pull down menus to allow non-programmers to quickly develop realistic animations.

Proof from Wolverine Software Corporation is a low priced, high performance animation program. Originally designed to work with GPSS/H, Proof's power and flexibility appeal to analysts using other simulation software products. A family of Proof Animation products currently exists including:

- Proof 3D
- Proof Professional with unlimited memory for large, complex animations
- Personal Proof which has size and memory limitations
- Student Proof Animation intended for learning and classroom settings
- Run-time Proof intended for displaying Proof animations but lacking development
- Proof Demo Maker and Demo Viewer which allow animations to be distributed
- Proof for Extend which is a custom version of Proof for the Extend simulation product from Imagine That, Inc.

2.5 Bibliography

Arena Simulation Information, http://www.arenasimulation.com/, Retrieved June 26 (2009).

D.T. Brunner (Chair) "Easy to Use Simulation 'Packages: What Can You Really Model?" In Proc. 1988 Winter Simulation Conference (San Diego, Calif. Dec 12–14), SCS, San Diego, Calif., pp. 887–891 (1988).

D.T. Brunner and James O. Henriksen. "A General Purpose Animator," In Proc. 1989 Winter Simulation Conference, SCS, San Diego, Calif., pp. 155–163, (1989).

Imagine That and ExtendSim Information, http://www.extendsim.com/, Retrieved June 26 (2009).

R.W. McHaney, "Discrete-Event Simulators Optimize Industrial Operations," Personal Engineering & Instrumentation News, (Aug), pp. 57–62 (1989).

Minuteman Software Information, http://minutemansoftware.com, Retrieved June 23 (2009).

T.J. Schriber, Introduction to Simulation using GPSS/H, John Wiley & Sons, New York (1990).

T.J. Schriber and D.T. Brunner, "Inside discrete-event simulation software: How it works and why it matters," In Proc. 2008 Winter Simulation Conference, SCS, San Diego, Calif., pp. 182–192 (2008).

Simscript III Information, CACI Products Company, http://www.simscript.com, Retrieved June 23 (2009).

Ingolf Ståhl and Beliber AB, WebGPSS Information, http://www.webgpss.com, Retrieved June 21 (2009).

C. Standridge, "A Tutorial on TESS : The Extended Simulation System," In Proc. 1985 Winter Simulation Conference. SCS, San Diego, Calif., pp. 73–79 (1985).

Wolverine Software Corporation Information, http://www.wolverinesoftware.com, Retrieved June 23 (2009).

3 Applications of Simulation

There has been a long running debate that concentrates on trying to decide whether simulation should be defined as an art or a science. Those who believe it to be a science feel that statistics, mathematics, and computer science comprise its foundation and are the basis for this classification. Others feel that the skill of the modeling team, the creativity involved in developing the model, and the interpretation of the results all add up to an individualized art. In this author's opinion, there is no real way to scientifically structure a simulation to guarantee that its results are valid.

Instead, after the model has been coded and run, the outputs can be studied and compared with corresponding known values to determine suitability. If no known values exist then the modeler must rely on his instincts and the judgment of experts to make this determination. The creativity and instincts used are akin to an art. Much of the methodology involved in model creation and analysis are based on computer science and mathematical principles. Therefore, elements of art and science exist in modeling. Simulation best may be defined as a "soft-science" containing both.

Figure 3.1 depicts the spectrum extending from art to science and the simulation's place. In recent years simulation has been moving along the spectrum more toward the science end. Improvements in simulation languages and the advent of simulators have removed some of the need to be as creative and innovative. Much of the uncertainty in model creation has also been eliminated through the development of application specific languages.

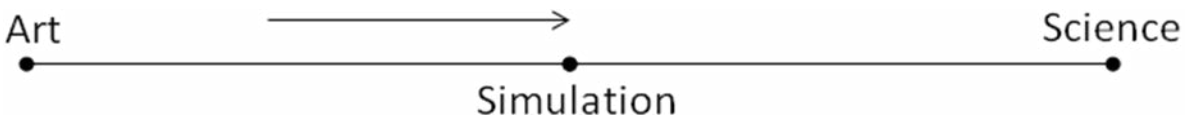


Figure 3.1 Simulation is Becoming More Scientific

3.1 Why Use Simulation

Numerous explanations exist for reasons behind the phenomenal growth computer simulation has experienced both in terms of application areas and in the number of available software products. Among these reasons are:

1. Improvements in computers: The first simulations were done on large, room-sized mainframe computers. These computers relied on card decks and operated most often in the batch mode. Not many people had access to these mammoth devices. In the last thirty years, computers have been reduced in size and cost considerably. Equivalents of the mainframes that used to occupy large rooms are now carried around in briefcase sized packages. Computers have moved from research laboratories and can now be found on practically every desk in all industries. The computing power of a single chip is fast becoming all that is necessary to run the most sophisticated commercial simulation software. This widespread availability and reduction in cost of computers has enabled simulation to prosper.
2. Improvements in simulation products: Thirty years ago, most simulation work was done using assembly language, FORTRAN, or other high level languages. Development time was much greater, as was debugging, statistic tabulation, and the reliability of results. This situation began to change with the advent of GPSS in 1961. Since that time, a multitude of simulation languages, analysis programs, animators, and pre-programmed simulators have become available. Much of the development time required to create a model has been eliminated through standard features found in these products. In addition to performing the necessary simulation functions, varying degrees of user friendliness are available. Simulation languages such as Simscript and GPSS/H appeal to practitioners with programming skills while non-programmers can enjoy the mouse driven menus found in many application specific simulators.
3. New opportunities for simulation education: Three decades ago, very few universities offered discrete event simulation classes. Today many universities offer simulation classes in engineering and business curriculums. In addition, private seminars and training sessions are available. These sessions are sponsored by simulation software vendors, consultants, and corporations with an interest in simulation and modeling. Other sources of simulation education are trade magazines, academic publications, conferences, societies, and books.
4. Increasingly complex and technical work environments: During the previous few decades the average work environment has changed from simple manual assembly lines to complex automated systems. Many repetitive human tasks have been replaced with robots and factory automation equipment. Conveyors, forklift trucks, storage and retrieval racks, as well as many other factory floor items are now routinely controlled by programmable logic controllers or computers. This new complexity has made factory output rates very difficult to predict and manage. Thus, the need for better analysis tools arose. New simulation techniques evolved to satisfy this demand and were able to remove much of the guess work and uncertainty in the work place.

5. Computer literacy among analysts and engineers: Computer literacy and use is nearly ubiquitous among professionals. Everyone has the ability to receive computer training.
6. Competition and tightening budgets – Another factor adding to growth in simulation use is an emphasis on lowering overhead costs, reducing labor requirements, and streamlining operations. Much of this has come about as globalization has increased competition and businesses compete on an international basis.
7. Realization of the benefits offered by simulation: As more industries recognize the benefits of simulation, investing in capabilities to use these tools becomes more important. Apparent economic advantages have prompted companies to invest time and resources into this area.
8. Industrial peer pressure: Organizations without simulation capabilities have found themselves at a competitive disadvantage. This was most apparent in industries, such as materials handling, where a simulation study accompanying a quotation would lend credibility to a proposed system and often become a determining factor when the contract was awarded. A situation of industrial peer pressure was created. Purchasers would inquire why simulation was not being used by their vendors and demand that some type of through-put guarantee be given. Presently, most materials handling system request-for-quotations require modeling be performed prior to purchase. Similar requirements exist in other industries. These forces have been a key factor in popularizing simulation.

9. Warm fuzzy feeling: Many companies have developed internal simulation groups to model in-house problems, proposed systems, and existing manufacturing processes. Presentation materials such as simulation animation packages have helped to sell internal proposals to management and create a corporate warm fuzzy feeling. "Seeing is believing" and many simulation packages available today place an emphasis on graphics and output.
10. Part of an effort to increase quality: Another force related to simulation adoption is a commitment to quality improvement processes. The success of these philosophies has been demonstrated in many industries with large productivity gains and improved profitability. A major precept of quality is prevention of problems. By creating a model of systems to be designed, purchased, or installed, costly mistakes can be avoided and installations can be done right the first time.

3.2 Simulation as a Design Tool

Depending on the application area, simulation can be used for different purposes and to answer a variety of questions. It is important to know what information the simulation is to provide at an early stage in its development. This ensures sufficient detail is incorporated into the program and helps to prevent developing inappropriately scaled models.

In addition to application area, the stage in the simulation process determines what questions should be asked. For example, early in the study answers to broad scale questions may be sought. General feasibility is often determined with a simplified version of the model. If the system performs as desired in this stage of the project, then more detail can be added and different types of information can be sought. Simulations of the preliminary nature are often performed with spreadsheet programs, rapid-modeling tools or simulators. Table 3.1 provides examples of questions that may be asked at different stages in the simulation process.

1) Preliminary Stages (global questions of feasibility): Do we have sufficient plant capacity? Is this type of system feasible for use in our application?
2) General Strategy Questions: Would an Automatic Guided Vehicle system or Monorail be more appropriate? Should a Just-In-Time inventory scheme be employed?
3) Specific Design Questions: Can the chosen conveyor move enough parts to station A per unit time? Are these work cycle times fast enough to keep up with the product flow? Should a single queue feed the four stations or should one queue feed all four?
4) Fine Tuning: What is the optimum number of Automatic Guided Vehicles to run the system? How may queue positions would be best at this work station? Should employee coffee breaks be staggered?

Table 3.1 Simulation Questions

Questions such as these are both application specific and project stage dependent. It is important know a simulation's purpose and keep sight of its goals throughout the process.

3.2.1 Types of Simulation

During the early stages of a simulation process when broad conceptual questions are being investigated, it is advantageous to use fast, less detailed modeling techniques. When these global concerns are resolved, the model can be refined with more detail until a complete understanding of the problem is gained.

A less detailed first run simulation is commonly called a concept simulation, rough cut simulation, or rapid model. This type of model tests overall concepts without the addition of finer details. Not only does a concept simulation save time, it also gives the modeler insight regarding problems needing more work when the detailed analysis is conducted. The model simplifications often make it possible for changes to be made faster and with less effort. This allows different alternatives to be tested in a short amount of time.

Concept simulation does have pitfalls. In order to simplify the model, detail is sacrificed. This means that the results can be inaccurate. Depending on what simplifications are made, models can be made completely invalid. Crucial factors should never be oversimplified.

Concept simulation is most applicable in situations where similar systems are modeled frequently. For example, the manufacturer of monorails might use concept simulation to provide potential customers with a general system picture. Since this particular vendor has simulated and built many monorails in the past, he can be fairly confident that he has not detrimentally over-simplified the system. A budget price can then be calculated without ever performing a detailed simulation. If the customer likes the concept and the price, the simulation can move to a new stage of development where additional detail is added.

Detailed Simulation generally follows the completion of a concept simulation. Here, the model is ready for the addition of finer details. There are two ways of accomplishing this. The first method is to rescale the existing concept simulation. The second is to develop an entirely new model, using what was learned in the concept simulation. It is common to choose the second alternative for several reasons. The concept simulation in many cases is written using a spreadsheet, simulator, or rapid modeling package. The development of a more detailed model will often require a simulation language or more complex simulator be employed. This will provide the modeler with the necessary flexibility to enhance his simulation.

In this phase of the modeling process, the simulation analyst would work very closely with engineers, system experts or system designers. Actual measured distances, carefully calculated work cycle times, and scheduling algorithms would be utilized. Realistic stochastic probability distributions would replace the estimated deterministic times used in the concept simulation. Machinery down-times, repair rates, and other system inefficiencies would be added. The model would be run, tested, and either compared to an existing similar system or examined by experts for correctness. By the time the detailed simulation was complete, the entire system being modeled would be defined.

3.2.2 Operational Changes and Follow-up

After a real world system has been installed, parameters estimated in the model can be validated through measurement. For instance, a manual assembly operation expected to take three minutes may actually take four. The simulation should be updated to as theoretical values are replaced with actual measurements. If the model is maintained in this way, it can be used in the future to test the impact of desired (or undesired) changes to the system, to estimate daily production, or as part of a larger overall simulation.

3.3 Estimation of Simulation Time

Many simulation projects in industry are completed within the context of a job contract or departmental budget. Consultants, materials handling vendors, even in-house simulation departments are usually obligated to provide an upfront estimate of time the project will require. This makes an accurate prediction of the expected time frame essential. Since simulation is based on the analysis of processes that are unknown, potential problems, requiring additional analysis time, can easily arise. Therefore, an accurate estimate of time can be a difficult value to determine.

When simulation projects overrun their allotted time, usually one of the following scenarios is to blame:

1. Unexpected problems are uncovered with the model and then it is used to find solutions: A function of simulation is to detect unexpected problems and help test potential solutions. Whether large numbers of problems or even a single, complex problem are found, required simulation time can increase drastically. When completion time has been estimated for the simulation, the importance of recognizing potential problem areas of the system and the addition of time for these areas is essential. A solution to this common dilemma is to set an initial time frame for modeling the system as it has been designed. If problems are found requiring additional simulation time, a new end date can be determined. In this way, the end date would extend on an "as-required" basis.
2. The simulation customer wants "just one more scenario tested": A tendency exists to keep playing different "what-if games" once a model has been created. This tendency can lead to creative and innovative enhancements to a system but it can also extend simulation project time and cause deadlines to be missed. When estimating time for a simulation project, areas that can be changed and "played" with should be identified as being experimental variables. Extra time can be added into the estimate up-front that takes into account the desire to try several different ideas after the initial model is complete. Once the estimated time is used up, no more "what-if games" should be allowed unless extra time is purchased or approved.
3. The system is more complex than expected: Sometimes an initial inspection of a system does not reveal all of its hidden complexities. What may appear to be a straight forward process may in fact be very complicated. For this reason the estimator must be certain that he or she fully understands all aspects of the system to be simulated.
4. The simulation analyst lacks experience: Another reason for estimates exceeding their time frames is the lack of experience by the simulation analyst. On a first time project, extra time has to be included to allow the analyst to learn the simulation language, hardware, and experimentation methods. It is generally not possible to build a first time model without experiencing some delays due to the learning curve. Once the simulation team has a few projects under their belts, these delays will no longer be a factor.
5. Too much non-essential detail is included: The construction of a simulation requires that the analyst know what information is being sought. By having a question to answer, the system can be broken into essential and nonessential subsystems. This is called scaling. The key subsystems can be modeled in detail and the other subsystems can be modeled in less detail. By taking this approach, the model will remain realistic while conserving development time and effort.

The following example illustrates breaking a system into essential and nonessential subsystems. In this scenario, the assembly department in a large manufacturing plant would like to speed up operations by automating a welding operation with a robot. The system will consist of a forklift truck dropping the work piece on a conveyor. The conveyor moves the piece to the welding robot. When the operation is complete, the work piece moves on to a manual station for grinding and deburring. It is then removed from with a crane and placed into a crate for shipping. See Figure 3.2.

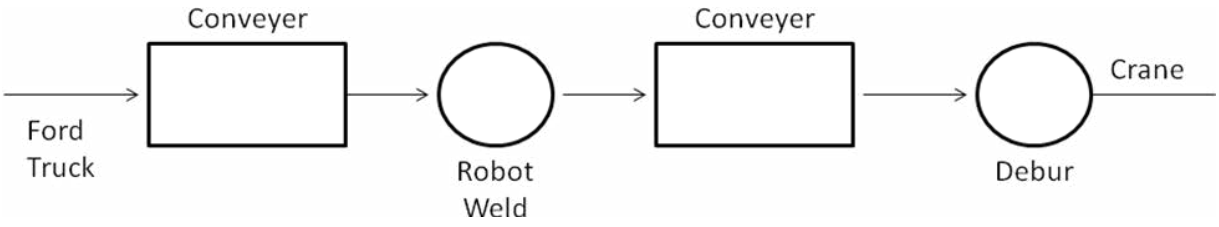


Figure 3.2 Automated Welding System

The goal of the simulation is to determine if the robot's work cycle time will be fast enough to process twelve work-pieces per hour. The system to be simulated can be broken into these components:

1. Forklift truck
2. Conveyor to robot work station
3. Robot weld operation
4. Conveyor away from robot work station
5. Grinding and deburring operation
6. Crane move to shipping crate

The most essential subsystem to be modeled is the robot weld station. It should be done in detail. The conveyor system to and from the robot may have a significant impact so it also should be modeled in detail. The grinding operation is manual. Workers can be added if needed, so it does not need to be modeled to the same level of detail. The crane operation will not be modeled in detail because unmoved work pieces do not slow down the grinding operation. See Figure 3.3.

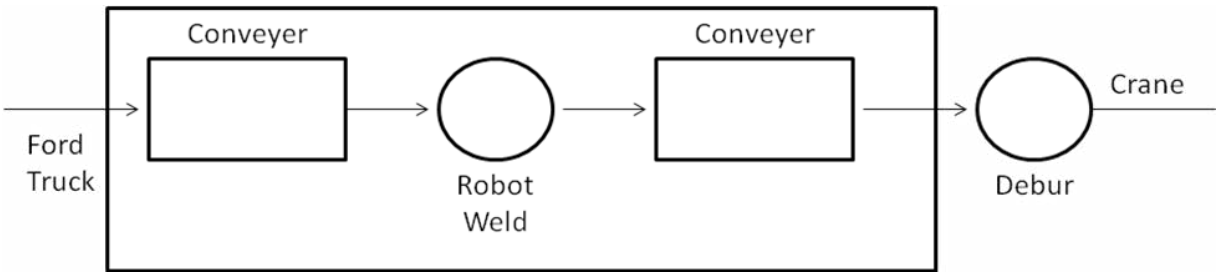


Figure 3.3 Automated Welding System Detailed Area for Simulation

By breaking the system into subsystems and defining the necessary level of detail, the goals of the simulation can be met and time can be conserved.

In the business world or manufacturing sector, it is important to be able to predict the length of time creating a simulation model will take. Some factors to consider when doing this are level of detail desired, size of simulation, project objective, and simulation software language being used.

3.4 Methodology for Manufacturing Simulations

In the manufacturing sector, a simulation methodology incorporating both concept and detailed simulation is frequently used to facilitate the design, selling, and installation of materials handling and other industrial equipment. This methodology can be summarized in the following six steps developed by Wesley Cox (see Table 3.2):

Step #	Step Name	Description
1	Concept Simulation	The creation of alternate models to test the most promising of the possible materials handling solutions.
2	Acceptance of Concept	The prospective customer accepts what she/he feels is the best alternative. Price, functionality, and through-put requirements are considered.
3	Engineering Design	The simulation is redone by adding details and final design parameters.
4	Operational Changes	Final changes are made to accommodate the changing requirements of the customers. The changes are simulated and tested for cost effectiveness.
5	Final Report	A final simulation report is provided and, if contracted, a system animation.
6	Maintenance	If desired, the simulation is maintained throughout installation of the system. Any changes or required modifications are incorporated into the model and the impact on the system is tested.

Table 3.2 Manufacturing Simulation Methodology

3.5 Forcing Completion of Design with Simulation

Due to the nature of most simulation studies, model creation will be among the first few activities slated for completion in the broader life cycle of a development or construction project. Simulation is a process which requires that all aspects of a system be considered and planned. To create a model, information must be gathered from a variety of sources and compiled for use. This gathering process will often bring different people and ideas together for the first time. Communication between persons with responsibility for the different parts of the system is facilitated.

Subsystem interface points will be discussed and obvious problems will be eliminated. Common goals will be set. Aspects of the system which were not previously considered may come to light. The formulation of creative solutions and innovative ideas may evolve. By the time the simulation is ready to be coded, very few unknown variables will remain.

3.6 The Simulation Decision

"To sim or not to sim?" that is the question.

In many companies the decision to use simulation is one that requires careful thought and an investment of thousands of dollars. For the company that has a simulation department in place, the question takes on a different meaning. It means 'should this particular problem or system be modeled?' Since simulation is expensive and time consuming, its use must be framed within the proper context. The possibility of over-modeling exists. Simulation should not be used until other methods of analysis have established the system being studied is viable. The following steps can help prevent the problem of over-modeling:

1. Make sure conceptual designs are complete and viable: An important step in system development is the forming of overall concepts. Simulation can be used to test a concept or it can be used to compare different concepts. Before simulating, ensure concepts are complete and can be used if found to be workable.
2. Look for obvious bottlenecks: Don't simulate a system that obviously can't work. Many bottlenecks can be spotted before the modeling effort begins. These evident problems should be resolved prior to simulation so extra effort and costly delays are avoided. While a system is being simulated, the analyst should continue to look for bottlenecks for the same reason. The sooner a problem comes to light, the sooner it can be dealt with.
3. Make sure a simple mathematical solution does not exist: Before the modeling process begins, use mathematics to answer questions. Not only can many situations be understood deterministically, time spent and costs may be lower.

3.7 Make It Work Vs. Does It Work

A simulation project will be conducted differently based on its overall objective. Two common objectives are: answering the question "does it work?" and taking on the task of "making it work." See Figure 3.4.

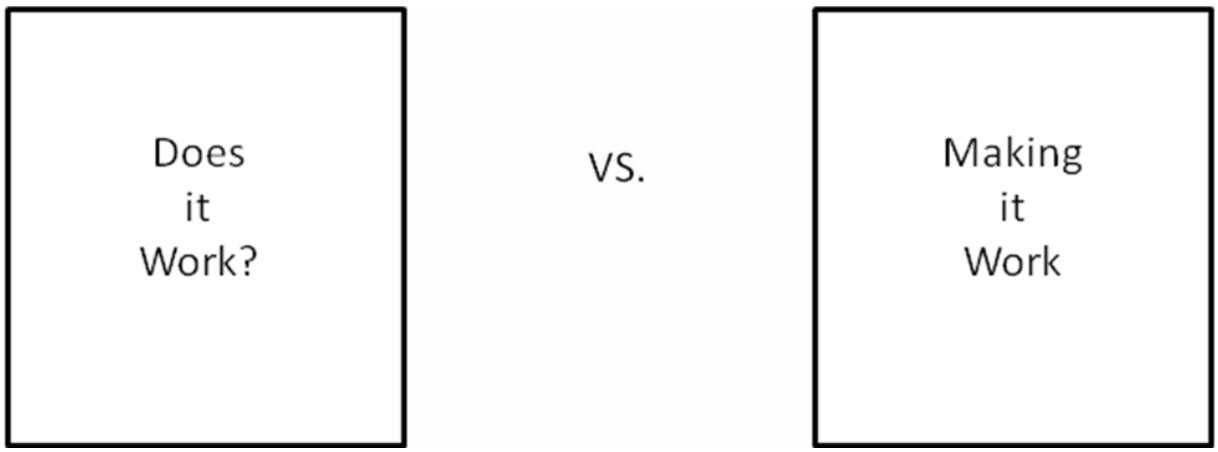


Figure 3.4 Simulation Project Objectives

Does It Work? – This project objective is clear cut and implies a model is to be used to evaluate a concept and determine if it meets a performance objective. Often the "does it work" question is used to test various alternatives for comparison. The simulation results will provide a yes or no answer to the problem being modeled. The "does it work" objective is usually easier to estimate and involves little "what-if" scenario testing.

Stamping4Success Simulation #1
A stamping machine that was purchased several years earlier became a point of controversy at Stamping4Success. Not enough work pieces were being processed each day to meet the requirements of a new contract. Management pointed its finger at machine operators saying they weren't doing their job fast enough. The operators in turn pointed their fingers at the machine saying it couldn't stamp the parts fast enough. A stalemate developed. As a means of resolution, a simulation analyst created a model of the process. The objective was to determine if the machine was fast enough to perform the required work. In other words, the model had the "does it work" objective.

Making It Work – The "making it work" objective is used when a problem exists and the simulation analyst must experiment with the system until a solution is found. Estimating an accurate completion date for models with this objective can be very difficult. It is not known prior to the simulation process how many different scenarios must be investigated before a solution is discovered. The Stamping4Success example is continued.

Stamping4Success Simulation #2
The initial simulation was completed at Stamping4Success and it showed management was correct on their assumption that the machine was fast enough to process enough work pieces to make the new contract's quota. But, the model also revealed an additional problem. Although the stamping machine was fast enough to process the parts, the system delivering the work pieces to the operators was not fast enough to keep the machine busy. A decision was made to authorize a second simulation study. The new model would have the objective of "making it work". The simulation analyst, working with a plant engineer, was told to keep trying different delivery schemes until a workable solution was found.

3.8 Optimizing and Developing Solutions

A widespread misconception about simulation involves its output information. Often, people new to simulation will imagine that their model will not only solve the problem but also provide an optimal solution. This is not necessarily, nor even ordinarily, true. Simulation can be used to optimize system performance but only in conjunction with the careful design of the proper experiments.

In many simulation studies, especially within the context of a manufacturing environment, only several feasible alternatives exist. These alternatives can be ranked subjectively in order of preference, cost, and performance. The most desired solution is simulated first. If the system does not perform at the required level, the next alternative on the list is tried and so forth until a workable solution is discovered. Use of this methodology is not scientific and will not optimize performance, but it is very practical and will result in an acceptable outcome.

If a list of possible solutions is difficult to rank and an optimal result is desired, common methods of experimental design can be employed. The first step is to identify the model's factors. Factors are defined as input parameters and structural assumptions. These factors can be quantitative and represent a specific numeric value. They may also be qualitative, representing the structural assumptions of the model. An Automatic Guided Vehicle programming algorithm is an example of a qualitative factor.

Whether the AGV picks up the closest or oldest load is a structural assumption used in the model. Factors can also be identified as either controllable or uncontrollable. Controllable factors are ones which can be changed or managed to help improve system performance. The number of AGVs used in a warehouse simulation is a controllable factor. Uncontrollable factors are ones which can't be changed within the context of the model. An example of an uncontrollable factor is the AGV's top speed.

The second step in optimizing a solution is to use viable combinations of the controllable factors as inputs in different simulation runs. The model can be run and the output performance measured. This output is termed the response. The varying responses can be screened based on meeting a minimum acceptable performance level. Experiments falling below this level will cause that potential solution to be dismissed.

The final step in optimizing the solution list is to employ some method of analysis to rank the list of acceptable factors and their responses. It may be possible to do this through inspection, by using methods of linear programming, or based on some other criterion. When the ranking process is complete, the model with the best combination of inputs and outputs will be chosen as optimal.

3.9 Genetic Algorithms

Other techniques such as incorporating genetic algorithms into a model also can be used for optimization. The genetic algorithm is a search algorithm based on the mechanics of natural selection and population genetics. These algorithms combine machine-learning with the principles of survival of the fittest. While genetic algorithms are not developed specifically with optimization in mind, minor modifications often result in adequate optimizers. Genetic algorithms work with populations of solutions and attempt to guide a search for improvement through survival of the fittest. Each potential solution is compared to a fitness function and is deemed eligible to propagate or is terminated. Over time, stochastic processes encourage the fittest solutions to emerge.

The genetic algorithm relies on a creation paradigm resembling biological reproduction. Artificial creatures comprised of binary strings are created in software. The combination of bits encoded in each string is a potential solution to a problem. These strings are stored and manipulated in computer memory.

The initial population of strings is created randomly. This population is compared to an objective function. Variations in the individual strings result in varying degrees of fitness. The fitter solutions are given a higher probability of survival and thus have a greater chance of passing their characteristics to future generations. Surviving strings form the basis for a new population. Subsequent generations are formed through reproduction, crossover and mutation. Reproduction means a string is directly copied from one generation to the next. Crossover combines characteristics of two or more strings much in the same way that a child inherits genes from his/her parents. Mutation involves small local changes such as the flip of a bit. Reproduction and crossover are the driving forces behind continual improvement of the population while mutation allows wider exploration for new solutions.

3.10 Ethics in Simulation

All designers, managers, programmers, analysts and engineers will confront issues that require ethical judgment. Occasionally this issue will be related to a simulation study. Often, a great deal of trust is put into a simulation study. The analyst must produce answers to questions that may involve large expenditures, vendor selection, designer prestige, and whether a system may even be purchased. Due to the nature of simulation studies, an analyst may be faced with a dilemma similar to the following examples:

Scenario #1:
A project manager says, "If this simulation proves that our new concept will work, the buyer will select us as vendor for sure. This contract could mean millions in new revenue and a dozen new jobs. Plus, it could mean a big raise for you and a great commission for me." After completion, simulation proves the system falls short of desired performance. The project manager is informed and says, "Find a way to make the system work by Friday." Friday arrives and the system marginally falls short of production requirements. As a simulation analyst do you create a report saying it works (and hope a solution comes to light prior to system installation) or do you stand by the model's results and report that it won't work knowing that your company's bid will not be selected?

Scenario #2:
The designer on a materials handling project would like to persuade her system team to agree with her conceptual idea regarding system operation. She asks you to "color" the simulation's output data to give her argument additional weight when the concept is presented. "It won't really change anything," she says. "It will be easier to justify my idea and get the team onboard a little faster."

Scenario #3:
A salesman reported to the simulation department that all other vendors bidding on a hotly contested job proposed eight robotic work stations be used. An in-house simulation study revealed that only five were required to produce the number of parts needed by the potential customer. The salesman suggested creating a simulation report that recommends seven stations in order to make the sale larger. The bid would come in at a lower price than the competition but extra profit would be quietly made.

Scenario #4:
A potential customer's rush order is waiting for the completion of a simulation study. After getting the base model running but before validation and proper output analysis can take place, the project team leader demands a report. In spite of the simulation analyst's objections that model validation and proper output analysis will require several more days, the team leader replies that he doesn't care and wants whatever is complete now.

The following list is advice concerning ethics in simulation. It is compiled from the opinions of several seasoned simulation analysts.

1. Simulation should never be used to misrepresent data.
2. Simulation should never be used to mislead or as a means of false justification.
3. The simulation analyst should always conduct experimentation objectively.
4. The simulation analyst should always be truthful in reporting.
5. The simulation analyst should be careful to avoid allowing her or his optimism or pessimism color reports.
6. If a mistake is made, take immediate action to rectify the problem. Never cover it up or hope it remains unnoticed.
7. If a system marginally doesn't make the required production rate, report the results together with an error margin and participate in a decision of whether the results are close enough to warrant the risk of implementing the system.
8. Ensure results are accurate through verification and validation activities.
9. Work closely and openly with the users of your model.

As a simulation analyst, you never want to be in a position where you must explain why your "supposedly valid" model meets production when the real-life system is found not to work. Fixing the system after installation is very expensive. By adhering to a sound code of ethics, a simulation analyst is able to ground themselves with a set of guidelines that will help make difficult decisions. Although the analyst's actions may not always be popular, they will be consistent and defendable. In the long run, your peers will respect your professionalism and long term view of organizational quality.

3.11 Bibliography

P.A. Bobillier, B.C. Kahn and A.R. Probst. Simulation with GPSS and GPSS/V. Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1976).

W. Cox, "Methodology for Manufacturing Simulation", Materials Handling Engineering, May, p. 76 (1989).

J. Holland, Adaption in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, Michigan (1975).

R.W. McHaney, Computer Simulation: A Practical Perspective, The Academic Press, San Diego, California (1991).

R.W. McHaney, "Integration of the Genetic Algorithm and Discrete-Event Computer Simulation for Decision Support," Simulation, Vol. 72, No 6, (June), 401–411 (1999).

N.J. Radcliffe, "Non-linear Genetic Representations," Parallel Problem Solving from Nature (R. Manner and B. Manderick, Eds), Volume 2, Elsevier, Amsterdam, 259–268 (1992).

C. Trunk, "Simulation for Success in the Automated Factory," Materials Handling Engineering, May, 64–76 (1989).

4 Starting a Simulation the Right Way

A computer simulation project is more than purchasing modeling software, plugging in a few values and expecting an answer to a question to pop out. This chapter considers the broader picture that simulation encompasses by developing a methodology for starting the process correctly. However, before that is discussed, a failed project is examined.

Case Study: The Wrong Way

A young engineer, fresh out of college and ready to take on whatever challenges her new job had to offer, was assigned the task of simulating a proposed materials handling system. The system would be installed in a large engine plant having thirty manual assembly stations. Engine blocks would enter the system and be moved from station to station via automated guided vehicle (AGV). At each assembly station, an AGV would drop off the engine block together with a part tray. The station operator would perform the required work then press a button to signal for engine removal. An available AGV would be dispatched to the station and the engine would move to its next stop in the assembly operation. The concept seemed to be workable in theory, so a simulation was initiated to validate the concept.

This was one of the first simulations the young engineer had developed and so she was careful to use a text book approach to the project. She was given an objective by the senior system engineer of 'see if it works'. The next step was to gather information about the system. She obtained work cycle times, a plant floor layout, information about the factory breaks and lunches, shift information, AGV speeds, AGV loading and unloading times, system priorities, and other operational data. Feeling confident this data was adequate, she commended coding the model.

Model development was more complicated than the young engineer had anticipated. By the time the simulation was written, its scheduled completion time had all but elapsed. The initial run of the system demonstrated the required number of engines could not be manufactured. Tempers flared under the tight time constraints and the simulation was given a new objective of 'making it work'.

Feeling pressured, she (with the help of a seasoned analyst) identified thirty-two controllable factors in the model. These were altered and tried in different combinations but desired throughput could not be obtained. Finally, a solution was found but it would require assembly workers to operate at inhumanly fast speeds at certain stations. Management at the engine plant felt no more time extensions could be granted and the system was installed.

For the first year, the system was slated to operate at half capacity. When the time arrived to ramp up to full operation, as the engineer had determined earlier in the simulation, the system was unable to meet required throughput. The simulation was brought out for further review but it only confirmed the real-world situation. The model was updated and different ideas were tried. Eventually, a costly solution was implemented by adding automation hardware to parts of the system.

Simulations can go terribly wrong, particularly if the appropriate infrastructure is not carefully developed to support the modeling effort. Few simulation analysts can report that all of their models are flawless. The previous example analyzed a real simulation project that became disastrous and the steps taken to revive it. Although the model had been accurate, the simulation received criticism and blame for the problems. Several pitfalls can be identified by looking at the process in retrospect.

- First: The simulation analyst was inexperienced. She lacked practical knowledge of the system's operation and this slowed initial model development.
- Second: The simulation 'team' members didn't participate adequately. Although the model demonstrated problems did exist, time was not taken to solve these problems as a preventative measure. Rather, the system was installed and then problems were tackled at a much higher cost.
- Third: Goals changed mid-simulation effort. The original simulation was meant to discover if the would work. When production fell short, a new goal of 'making it work' became the directive.

The problems pointed out by this case study could have been avoided if the simulation effort had been taken more seriously. If preventative action had addressed system inadequacies early, costly corrections and time delays could have been avoided.

A recent study of discrete event computer simulation projects revealed unsuccessful efforts are characterized by high costs, model size constraints, and slow software. In contrast, successful projects are characterized by teamwork, cooperation, mentoring, effective communication of outputs, high-quality vendor documentation, easily understood software syntax, higher levels of analyst experience, and structured approaches to model development. While these findings should not come as a surprise, the importance of having a structured methodology for approaching a simulation is apparent. The next sections of this textbook provide a detailed view of an approach to help enable simulation project success. The process used in this textbook is representative of many possible computer simulation project life cycles that have been used in practice and documented in simulation textbooks and journal articles.

However, remember each simulation project is unique and specific requirements may be better served through customizing the steps discussed. In general, the simulation project life cycle described in this book consists of six major steps, each having multiple components. The major steps being investigated are:

1. **Intelligence Phase.** The precursor to any solution is full development and an understanding of the underlying dilemma or problem.
2. **Managerial Phase**. In general, this step requires interaction with management and other non-technical staff of the organization to acquire necessary resources and support along with the formation of a simulation project team.

3. **Developmental Phase**. During this phase, the simulation model or models are created. System design, detail design, and coding all take place and rely on the interaction of the analysts and other members of the simulation team.
4. **Quality Assurance Phase.** While the model is being coded in complete or at least in prototype form, the analyst must ensure proper validation and verification. The ideas of testing and completion or integration are also important here. It may be necessary take testing one step further and begin development of face validity through interaction with end-users and management. Generally, quality assurance accompanies the entire modeling lifecycle.
5. **Implementation Phase.** Model use begins and decision support activities take place.
6. **Operations, Maintenance, and Archival Phase.** Development is essentially complete and only maintenance tasks remain. This phase may be substantial if the project results in a simulation maintained for repeated use

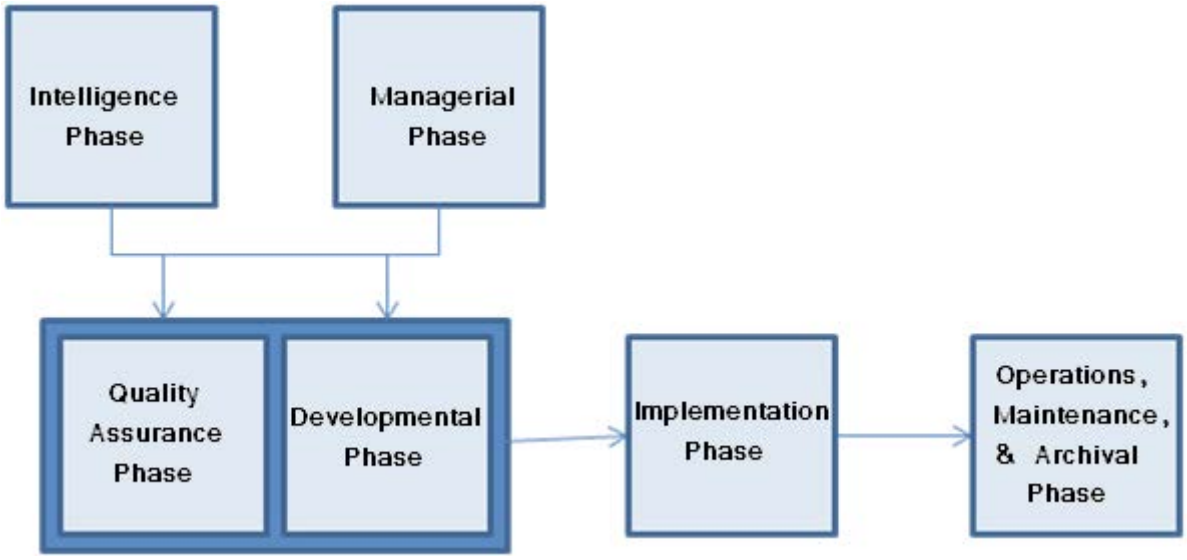


Figure 4.1 General Simulation Lifecycle

A simulation project can be a complex organizational undertaking. In general, to believe a simulation project starts and ends with coding a model would be a mistake. Much more is involved. In many instances, modeling is one small component in a much larger simulation project life cycle. As stated previously, a simulation project is inclusive of a wide spectrum of activities ranging from system and problem definition to ensuring logic developed in the model can bridge the gap between model and real world and be used in actual system implementation. The following sections provide a closer look at the mechanics of conducting a professional simulation study.

4.1 Intelligence

The intelligence phase of the simulation life cycle involves understanding the environment and determining problems to be solved. In general, this phase consists of the simulation analyst or potential simulation customer discovering situations that require modeling. The simulation analyst should emerge from this phase with knowledge that a problem exists and should have at least a preliminary understanding of the problem's nature. Often, problem definition and feasibility are assessed at this point in time.

4.1.1 Problem Definition

In order to ensure simulation project success, several events need to occur before any other work begins. First, the customer needs to define objectives for the study. The objectives for the simulation describe what questions the simulation needs to answer. In other words, the precursor to any solution is a full understanding of the underlying dilemma or problem. Parameters for the problem as well as broad scale constraints need to be defined. A simulation project may be driven by a broad statement such as 'The Denver Chamber of Commerce has projected that incoming passenger flights will increase by 50% over the next 18 months." Definition of the problem operationalizes the statement and looks at the system environment to clarify goals for analysis.

For example the problem definition may become:

Can the existing baggage handling system handle the project increase in passenger arrivals and departures?

Of course, this question could rapidly be replaced with a related:

What needs to be changed in order to accommodate the expected increase in passenger arrivals and departures?

These sorts of clarifications need to be made early in the simulation life cycle.

4.1.2 General Feasibility

Together with problem definition, it is common to conduct a preliminary feasibility study at this point in the simulation life cycle. The feasibility study may be focused on various characteristics of the project and may seek to answer questions such as:

Will a model provide an answer to the question?
Can an accurate model be developed?
Can model development be accomplished within the desired timeframe?

Table 4.1 Feasibility Questions

As with any general simulation software development effort, feasibility may be assessed in a variety of areas. One model for feasibility is TELOS which stands for appraising technical, economic, legal, operational, and schedule feasibility.

4.2 Managerial Phase

A means for gathering information needs to be established. This requires obtaining support from management, domain experts, system users, and other individuals in order to understand and document the system that will be modeled. In other words, simulation requires teamwork and in most cases cannot be done by a lone individual.

The managerial phase facilitates organizational involvement in a simulation project. A number of overhead tasks are accomplished here including development of proposals, securing a budget, acquiring managerial support and forming a project team. Often, the development of a simulation proposal is used to organize effort and ensure all participants are working toward a common goal.

A project kick off meeting might be held at this point in time and expectations of higher level management might be discussed to avoid situations where the modeling effort degrades into a prolonged search for potential solutions to poorly defined problems.

4.3 Developmental Phase

After the objectives for a simulation study have been stated and resources allocated, modeling can begin in earnest. In general, modeling involves developing a representation of the system. A working definition for a system is:

A set of components or elements that are related to each other in such a manner as to create a connected whole.

The relationships between elements comprising the system are essential in predicting the behavior of the whole. If the system is reduced to a group of independent subsystems, each can be studied separately, but the true characteristics of the system may be lost. It is critical to maintain interaction between various system elements.

The orchestra shown in Figure 4.2 provides an example of a system with complex interacting elements.

Photo by Jordan Fischer Source: Flickr (Creative Commons)

Figure 4.2 Chicago Symphony Orchestra

Various subsystems consisting of groups of musicians playing similar instruments are related to each other in such a way as to create a connected whole. The stringed instruments can be viewed as a subsystem. So can the musicians playing only the first violin part. Percussion, brass, clarinets, flutes, cellos, French horns, and oboes can all be defined as subsystems. The various groups of musicians are dependent on their musical score and the direction of a conductor to achieve an overall goal. Each subsystem can be removed from the system, studied and analyzed, but it would become very difficult, if not impossible, to predict the overall behavior of the system in this way.

A system can be broken into its individual elements structurally. However, it cannot be broken into individual elements and still maintain its functionality. For example, a computer system can be studied in terms of its components such as microchips, wires, capacitors, resistors, transistors and electricity. If such a study was completed, the true characteristics of a computer would be lost. The crucial interaction between the individual components would not be present.

Within the context of a simulation project, the importance of viewing the system as a whole is quite evident. In order to maintain an accurate representation all interactions between the various components must be included in the model. In addition, the components and their interrelationships, the system's environment and boundaries need definition (Figure 4.3).

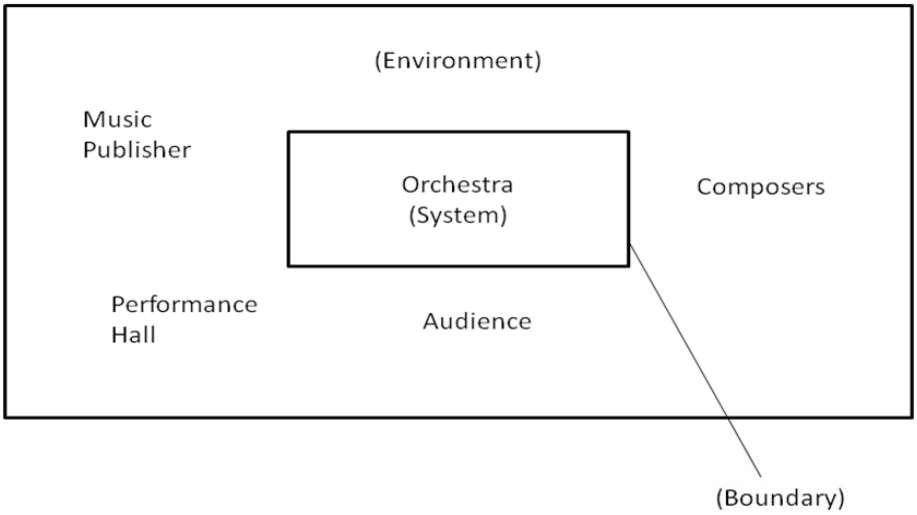


Figure 4.3 Environment and Boundary

The environment lies outside the system and has influence on its behavior but is not controlled by it in any way. A system's boundary is the line that separates a system from its environment. In the development of a simulation, influences of the environment must be taken into consideration and can be represented with constraints, rules, and events.

Environment and boundary can be illustrated using the example of an assembly line in a United Auto Workers (UAW) unionized plant. The system to be modeled is an auto assembly line. The system boundary encloses the materials handling system, the assembly operation, the assembly operators, and the product moving through the system. The environment which exerts influence on the system would be the set of union rules for coffee break times, the lunch break, shift change and holidays. The influence of these environmental factors can be represented in the model with various operational rules and events that impact model outcomes.

The division between system and environment is often a very delicate line to try to define. The aptitude for defining systems, subsystems, and environment is an acquired skill that might take years to refine.

In general, during the development phase the simulation model is created. System design, detail design, and coding all take place and rely on the interaction of the analysts and other members of the simulation team. A number of considerations must be included during model development.

4.3.1 Model Scaling

Model scaling is a determination of how much detail is required to emulate the real world system and still accurately provide the required information. Since a model is an approximation of a real world system under study, it is not always best to develop a full scale representation. If the system were modeled down to the finest detail, excessive time and energy would be expended with minimal gain in useful information. The law of diminishing returns can be applied to level of detail versus gain in knowledge. See Figure 4.4.

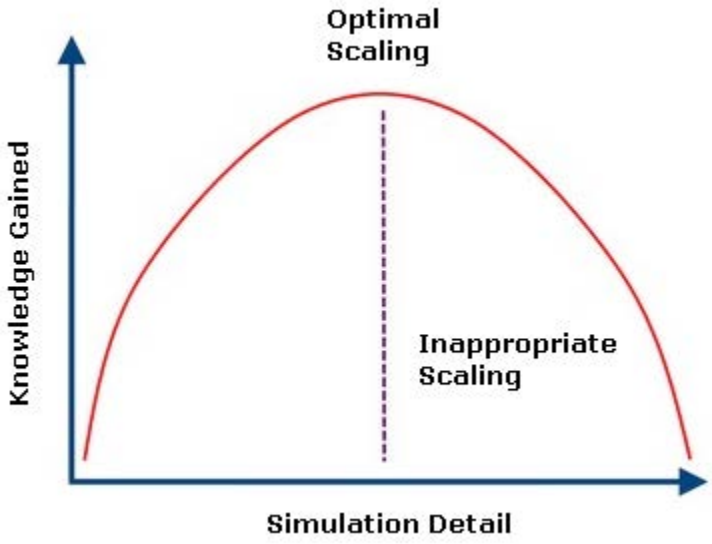


Figure 4.4 Scaling vs. Knowledge Gained

A recommended approach to model development is to incorporate the least amount of detail while still maintaining veracity of the simulation. This process, called scaling, is a subjective skill developed as a simulation analyst becomes more experienced. Different analysts will build their models with varying levels of detail with an ultimate goal of creating a valid and useful simulation.

Computer Manufacturing Company:
Serial Partitioning of a Model

A manufacturer of microcomputers needs to develop a model of its manufacturing and shipping operations. The analyst assigned to the project decides a single model would be too complex to develop with current time constraints and decides to focus on key portions of the system using serial partitioning. The first step in using this technique is to identify a natural break in the system's operation. After collecting data he finds all manufactured computers are stored in a warehouse and when ordered by a customer, are removed and proceed through the packaging and shipping operations. The point at which the computers leave manufacturing and enter the warehouse is a natural break. The interaction is one way and could be used as a place to break the model into two separate serial portions. The outputs from the manufacturing model will be used as inputs to a future storage and shipping model.

4.3.1 Model Scope

As part of system definition, model scope must be determined. Model scope determines the portion of the system to be represented by the model. Partitioning is a common technique used to do this and to help keep a model's size manageable. Partitioning involves studying individual subsystems and determining which can be represented by a single construct and which need to be modeled in more detail. This technique can work very well under certain circumstances, particularly when little interaction or interdependency occurs between the subsystems, caution should be exercised to avoid oversimplification.

Partitioning can be either serial or parallel. Serial partitioning will break a system into separate successive stages for modeling. The outputs from each stage will become the inputs for the next stage. If the interaction between the stages is unidirectional, serial partitioning may be adequate. If the interaction is bidirectional and complex, with a complex interdependency between the systems, partitioning should not be used. As an example, consider the following case:

Parallel partitioning can be used where a set of independent, identical processes are replicated a number of times. These processes can be modeled by a single entity having the capacity to simultaneously perform duties that in reality would occur in parallel.

Defining which parts of the system need to be modeled is very important to the success of a simulation project. Modeling in too much detail leads to excessive development time and unnecessary complication. Too little detail may result in a model which does not realistically predict actual world behavior. The analyst must use caution during this part of the simulation process to insure the model accurately reflects system operation.

4.3.2 Modeling Views

Development of a simulation requires a system be viewed in a manner that will facilitate the modeling process. Nearly all available simulation software products use one of three approaches: event orientation, process orientation or activity orientation. And, of course, use of a particular modeling tool or language may make this decision for the analyst by default.

Event Orientation – Using this modeling approach requires the simulation analyst to identify all events that might occur and determine the impact these events will have on the system. An event is defined as any instantaneous occurrence which may change the state of a system. The following example illustrates an event orientation approach used in defining a system.

K-State Credit Union Drive-through Bank Teller: A Second Look

After contemplating the use of the event orientation approach, Ms. Ida decided she would be better served to model the drive-through using the process orientation approach. After some research and careful thought, she developed the diagram shown in Figure 4.5. A customer entering the drive-through lane is considered to be a transaction and the teller, a permanent resource. The process routine describes the experience of the customer from the time he or she attempts to enter the lane until the time he or she completes interaction with the teller. Time passes while the teller performs the desired service. Time may also pass while the customer waits in queue for the teller resource to become available.

Process Orientation – A process is a time ordered sequence of interrelated events separated by passages of time. This sequence will describe the entire experience of a transaction as it moves through a system. When the process orientation approach is used, the modeler will define repeatable sequences of steps each representing an event or a series of events. These sequences of steps can be represented by a network or flow diagram. Objects, which may be required by a transaction during its movement through the sequence of steps, are defined as a resource. The following process orientation example takes another look at the K-State Credit Union drive-through.

K-State Credit Union Drive-through Bank Teller

Keri Ida, manager of the new K-State Credit Union branch opening near campus has to determine the number of tellers for the new drive-through service lanes. She thinks one or two tellers would be best and decides to develop an event oriented model to help make the determination. Ms. Ida identified the following key system events:

1. Customer Arrival - If the teller is occupied, then the person arriving will wait in queue. Otherwise, the person uses the services of the teller.

2. Customer Service Complete - Following completion of service by the teller, the customer leaves. The next person in queue can be serviced. If the queue is empty, the teller becomes idle.

Ms. Ida knew if either one of these events occur, the system's state would change. Having identified the key events, she was ready to begin coding his model using the event orientation approach.

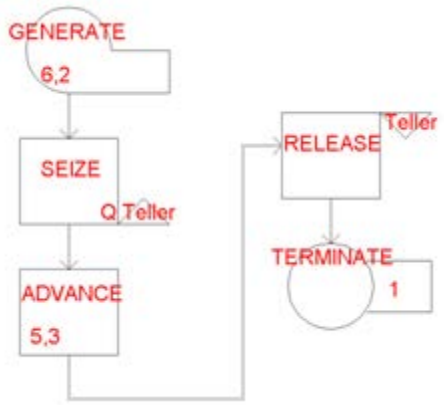


Figure 4.5 Block Diagram for Bank Model

Activity Orientation – The activity orientation approach is less commonly used. It involves describing a system in terms of the conditions necessary to either start or stop an activity. The beginning or conclusion of an activity will have an effect on the system state. Activities may require specified amounts of time for completion or may be brought to completion by meeting certain logical conditions.

4.3.3 Concept Model

Another early activity often utilized in a simulation project is the development of a general concept model. A concept or rough-cut model is a general model which can quickly produce an approximation of a system. Concept modeling is used to provide answers in the initial stages of a simulation indicating whether time and resources should be invested in a more detailed model. These rough-cut models can take the form of anything from a simple mathematical equation done by hand to a sophisticated network analysis done on a computer using specialized modeling software. Spreadsheet software is often the tool of choice for a concept model.

A great deal of work and effort in a simulation project often can be saved by taking a final look at system components from a quantitative viewpoint. If certain operations are mathematically infeasible, it may be desirable to postpone the detailed simulation and re-evaluate system design or even the premise of the project. The following example illustrates the use of simple mathematics to avoid an unnecessary simulation (See PikNGo: First Concept Model).

PikNGo: First Concept Model: Rough Calculations

PikNGo Packing and Storage uses a fleet of twelve forklift trucks to transport pallets of material to and from storage locations in their vast warehouse facility. Recently with labor disputes and higher fuel costs, upper management has decided to consider replacing the aging fork lift fleet with an automated guided vehicle (AGV) system. To help provide a preliminary estimate of the costs involved, PikNGo performed a rough set of calculations. Although a more detailed simulation is desired, the concept model was intended to indicate whether a more detailed simulation should be considered. Management had already looked at a budget and believed no more than a ten vehicle system could be afforded. Table 4.2 illustrates.

Results of Rough Calculations
Initially, the following numbers were used: Number of Loads Moved Per Day: NLD: To Be Determined Average Time to Move a Load: ATM: 12 Minutes Number of AGVs: AGV: No More than 10
TLPH = Total Loads per Hour = (60 minutes / ATM) * AGV
TLPH = 60 Min / 12 Min * 10 AGV
TLPH = 50 Loads

Table 4.2 Rough Calculations for PikNGo

PikNGo: Second Concept Model: Queuing Model with Spreadsheet

Currently PikNGo moves an average of 75 loads per hour so the rough calculations demonstrate the system falls short of the desired throughput. However, the model is not detailed and so rather than end here management decided to continue the analysis with more detail, this time using a spreadsheet queuing analysis model. The results shown in Table 4.3 were obtained. The numbers verify that only 50 loads could be moved in the desired time. A further run of the model indicated

Queue Station	AGV_Queue	
Arrival Rate	75	Number of Loads to be Moved per Hour
Service Rate/Channel	5	Number of Loads an AGV can Move per Hour
Number of Servers	10	Number of AGVs Available to Move Loads
Max. Number in System	***	
Number in Population	***	
Type	M/M/10	
Mean Number at Station	#NUM!	
Mean Time at Station	#NUM!	
Mean Number in Queue	***	
Mean Time in Queue	***	
Mean Number in Service	10	
Mean Time in Service	0.200000003	
Throughput Rate	50	
Efficiency	***	
Probability All Servers Idle	***	
Prob. All Servers Busy	1	
Prob. System Full	***	

Table 4.3 Spreadsheet Showing 50 Loads Moved with 10 AGVs

Queue Station	AGV_Queue	
Arrival Rate	75	Number of Loads to be Moved per Hour
Service Rate/Channel	5	Number of Loads an AGV can Move per Hour
Number of Servers	16	Number of AGVs Available to Move Loads
Max. Number in System	***	
Number in Population	***	
Type	M/M/16	
Mean Number at Station	25.95113945	
Mean Time at Station	0.346015215	
Mean Number in Queue	10.95113933	
Mean Time in Queue	0.146015191	
Mean Number in Service	15	
Mean Time in Service	0.200000003	
Throughput Rate	75	
Efficiency	0.9375	
Probability All Servers Idle	1.45347E-07	
Prob. All Servers Busy	0.730075955	

Table 4.4 Spreadsheet Showing 16 AGVs required to Move 75 Loads

Using a rough-cut or concept model prior to building a detailed simulation can provide the analyst with several benefits. These include:

1. Help identify problem areas early in the project.
2. Reduces the time and cost associated with unnecessary detailed simulation.
3. Allows rapid "what-if" testing.
4. Easy to use, inexpensive, and widely available. Spreadsheet software can be used in many instances.

4.3.4 Concept Modeling with Simple Spreadsheets

A simple spreadsheet can be used to rapidly prototype a model and to provide a general sense regarding a system's scope. This approach has advantages including speed of development but, if not used carefully and with a sense of its preliminary nature, can result in false expectations. The following example provides a simple spreadsheet example of concept modeling:

FastWay Delivery

FastWay Delivery Company decided to open a new branch office that would deliver bread and bakery items to grocery stores and service stations in its immediate area. Ten different routes were mapped out, each one consisting of varying distances and numbers of stops. An average speed in miles per hour was calculated for each route based on expected traffic congestion and legal speed limits. FastWay wanted an approximation of the number of delivery vans their new operation would require. They developed a concept model using Google Spreadsheet and these assumptions:

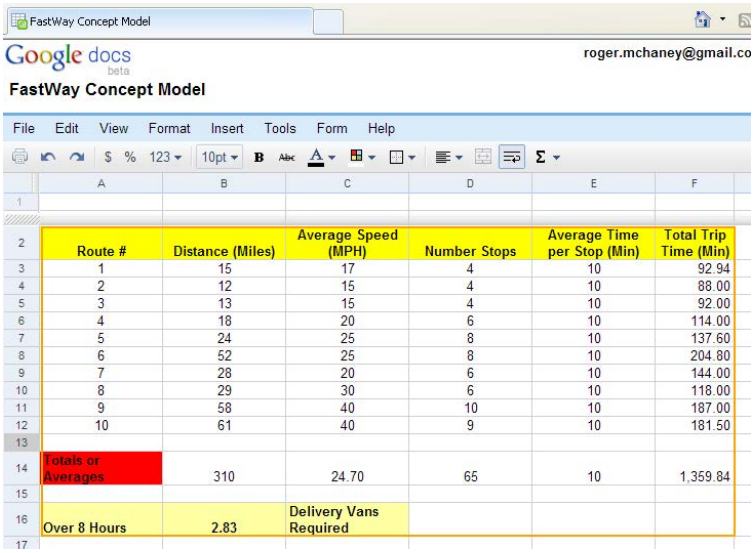
Assumptions

1) A route is a round trip leaving the bakery, servicing all stops, and returning to the point of origin.

2) A route can be taken at any time during the eight hour work day.

3) If a delivery van finishes its route before the work day is over, it will start the next route.

The spreadsheet that was developed is shown in Figure 4.6.



FastWay Concept Model

Google docs beta

roger.mchaney@gmail.co

FastWay Concept Model

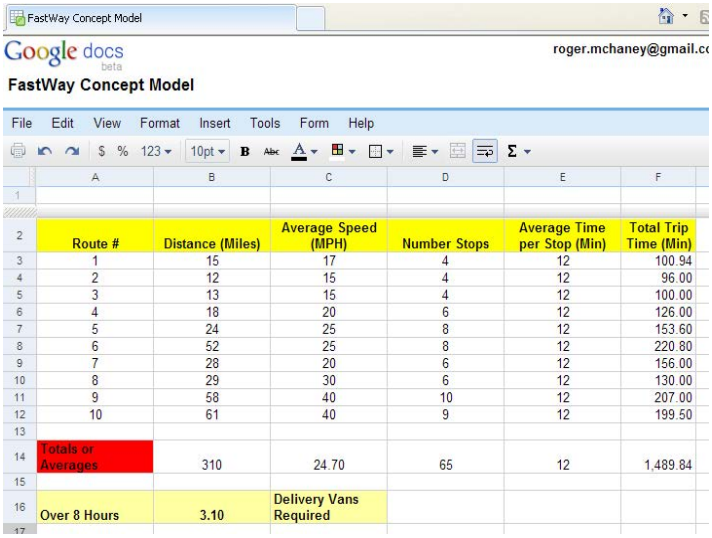
File Edit View Format Insert Tools Form Help

Route #	Distance (Miles)	Average Speed (MPH)	Number Stops	Average Time per Stop (Min)	Total Trip Time (Min)
1	15	17	4	10	92.94
2	12	15	4	10	88.00
3	13	15	4	10	92.00
4	18	20	6	10	114.00
5	24	25	8	10	137.60
6	52	25	8	10	204.80
7	28	20	6	10	144.00
8	29	30	6	10	118.00
9	58	40	10	10	187.00
10	61	40	9	10	181.50
Totals or Averages	310	24.70	65	10	1,359.84
Over 8 Hours	2.83	Delivery Vans Required			

Figure 4.6 FastWay Spreadsheet

FastWay was able to calculate a preliminary van count of 2.83 using the concept model that had been developed. Besides providing the desired data, management at FastWay was able to quickly alter certain parameters to experiment with the van count.

For instance, in Figure 4.7 the stop times were changed from 10 to 12 minutes. This increase indicated a 12 minute service time would cause the van count to rise to 3.10.



FastWay Concept Model

Google docs beta

roger.mchaney@gmail.c

FastWay Concept Model

File Edit View Format Insert Tools Form Help

	A	B	C	D	E	F
2	Route #	Distance (Miles)	Average Speed (MPH)	Number Stops	Average Time per Stop (Min)	Total Trip Time (Min)
3	1	15	17	4	12	100.94
4	2	12	15	4	12	96.00
5	3	13	15	4	12	100.00
6	4	18	20	6	12	126.00
7	5	24	25	8	12	153.60
8	6	52	25	8	12	220.80
9	7	28	20	6	12	156.00
10	8	29	30	6	12	130.00
11	9	58	40	10	12	207.00
12	10	61	40	9	12	199.50
13						
14	Totals or Averages	310	24.70	65	12	1,489.84
15						
16	Over 8 Hours	3.10	Delivery Vans Required			

Figure 4.7 FastWay Spreadsheet with Modified Stop Times

Since the calculated van count is close to 3 in both scenarios, FastWay may wish to explore the option of developing a detailed simulation incorporating randomness in the delivery times, van speeds, breakdowns, and stop times. The detailed simulation would provide a firmer van count and help them decide whether three or four vans should be acquired.

FastWay has another option. They may decide to purchase four vans knowing only three are required for the route work. The forth van can then be used as a spare, only working when one of the other three is in the repair shop or when extra deliveries are required. In this case, a detailed simulation wouldn't really provide any additional information that would impact the purchase decision. Therefore, the concept model may be the final step in the simulation process.

4.3.5 Concept Modeling with Advanced Spreadsheets

A wide variety of tools and add-ins are available to make concept modeling with spreadsheets easier. The models created for the PikNGo example were developed using the Queuing Analysis ToolPak 4.0 (Figure 4.8) available without charge from:

http://www.business.ualberta.ca/aingolfsson/QTP/default.htm

Figure 4.8 Queuing Analysis ToolPak 4.0

Another academic version of similar software (Figure 4.9) can be found at Paul Jensen's Website:

http://www.me.utexas.edu/~jensen/ORMM/index.html

Both of these sites provide add-ins to Microsoft Excel that can be installed and then used to develop a variety of concept models.

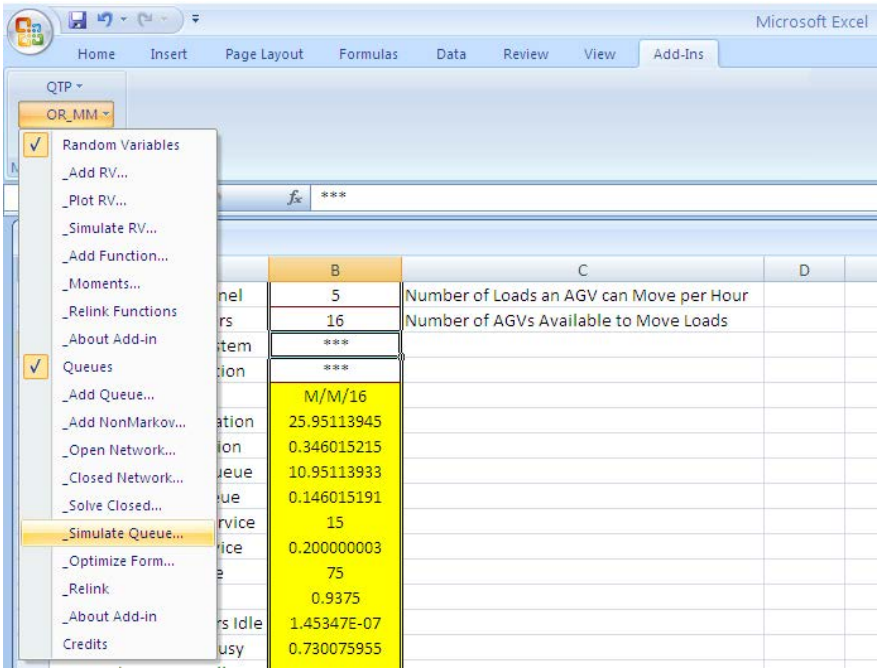


Figure 4.9 Paul Jensen's ORMM Queuing Add-ins for Excel

A variety of high quality commercial software is also available for conducting concept modeling. In many instances, a rough model quickly can be developed using a simulator or modeling software package such as ExtendSim or MedModel. Preliminary results can be quickly obtained and then used as the basis for a more detailed model after data is collected and system details studies.

Concept modeling can be either an intermediate step in a simulation process or an end in itself depending on the goals of the study. It can be used to provide quick, rough-cut information for preliminary decision making and should always be used as part of a complete simulation study.

4.3.6 Scientific Method Approach to Modeling

Nearly every discipline that employs research as a tool looks to the scientific method as a means of organization. The simulation field is no exception. The following steps make up the scientific method.

1. Definition of the problem.
2. Formation of a hypothesis.
3. Testing of hypothesis by experimentation.
4. Tabulation of results.
5. Drawing conclusions from results.

Table 4.5 Scientific Method

4.3.6 Model Inputs

Valid input data forms the basis for an accurate simulation. Remember the old saying "garbage in – garbage out?" A simulation will be no better than the information and assumptions it contains. Many projects have no chance for success because the input data used is not reliable (Figure 4.10).

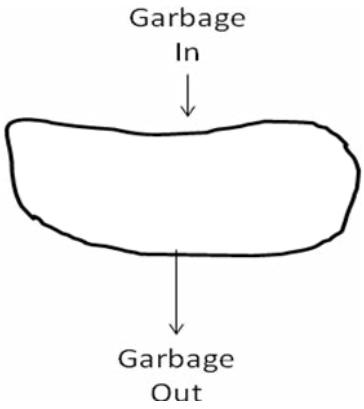


Figure 4.10 Simulation Reliability

Input data can take many forms. It can be quantitative or qualitative. Quantitative input data takes the form of numeric values. Examples would be vehicle speeds, number of cars passing a particular point on a freeway, number of tellers in a bank, acceleration rate of an automated guided vehicle or the process time at a welding station.

Qualitative input data is a more difficult parameter to measure. It may represent the algorithms used to perform certain logical operations. In the context of the PikNGo example, if a detailed simulation were developed, qualitative values would include the rules used to determine which load to move next or what happens when a vehicle goes out of service. Will other vehicles be reallocated? Will other changes take place? If an improper qualitative assumption is made, the model may not be accurate.

Input data can be obtained in many ways. Examples of several methods follow (Figure 6.5b):

Observation – A good method of obtaining input data is through observation. If a model were to be made of a busy intersection in a growing city's business district, a method of obtaining input data would be through direct observation and measurement of the traffic flows. Depending on the circumstances, data might be collected by people or through using computerized equipment. Once collected, the data could be used as a direct input to the model, although generally, it is better to use the data to select an appropriate input probability distribution.

Estimation – If the system to be modeled does not exist, it may be necessary to estimate input data. Although somewhat less than scientific, estimation may still help provide valuable insight concerning a system's operation.

Interpolation – If a similar system exists, its input data can be observed and then interpolated for entry into a model sharing the same characteristics. An example might be found in a machine shop setting. Five years ago a numeric controlled (NC) machine was purchased. It was able to keep up with required production rates until the present year when sales unexpectedly skyrocketed. In light of these changes, management had to make a purchasing decision. The manufacturer of the NC machine was contacted and recommended a new model which was able to process a third more material than the existing machine. By using data based on the NC machine in current use, interpolated values were derived and the expected operation of the new machine modeled.

Expert Opinions – Many times no true data is available and so expert opinions must be used in a model. For example, an engineer who has been responsible for the operation of a warehouse over the last twenty years may be able to provide data used in simulating proposed shipping docks. Although no real data can be collected, the engineer has an understanding of how the system will need to operate.

Projections – Model input data is commonly derived from future projections. For instance, a Japanese auto plant has enjoyed great success in marketing its products. They keep data depicting their growth rate over the last seven years. Next year they are forecasting a required increase in production of 21% to keep up with their expected sales. By using this projection, they are able to create input data to drive an assembly line simulation they maintain.

After input data has been collected, two approaches can be taken to utilize the information in the simulation. The first approach is to use real data as is. This technique is known as a trace simulation. The second approach is to fit a standard statistical distribution to the empirical data. In this instance, it is important to find a probability distribution so that random samples taken from it will be indistinguishable from the actual collected input data. Goodness-of-fit tests can be used to statistically validate this process. The following section explores several commonly used goodness-of-fit tests. After a probability distribution has been chosen, a random number generator or a pseudo-random number generator can be used to draw samples from the distribution and drive the simulation.

4.3.7 Statistical Methods and Input Data

Although empirical or "as is" data can be used for simulation input, the recommended approach is to fit a theoretical distribution to the data when possible (see Figure 4.11).

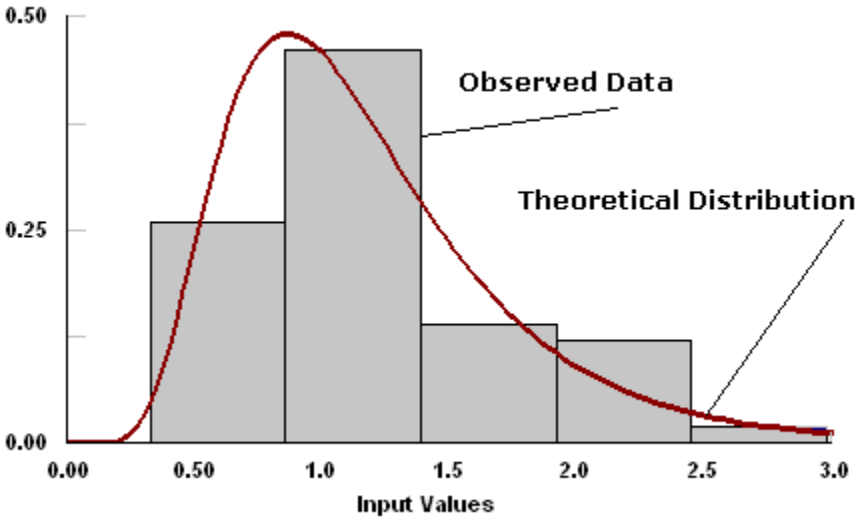


Figure 4.11 Fit Observed Data to Theoretical Distribution

If only observed data is used, then the simulation will never be exercised beyond that which was collected. In other words, extreme or unusual situations may never occur in the model. By using a theoretical distribution, the rare, large or small incidents will occur on occasion (Figure 4.12), thereby exercising the model beyond the limits of the collected data. Using a theoretical distribution in a model can make it more powerful and predictive than standard calculations or concept models.

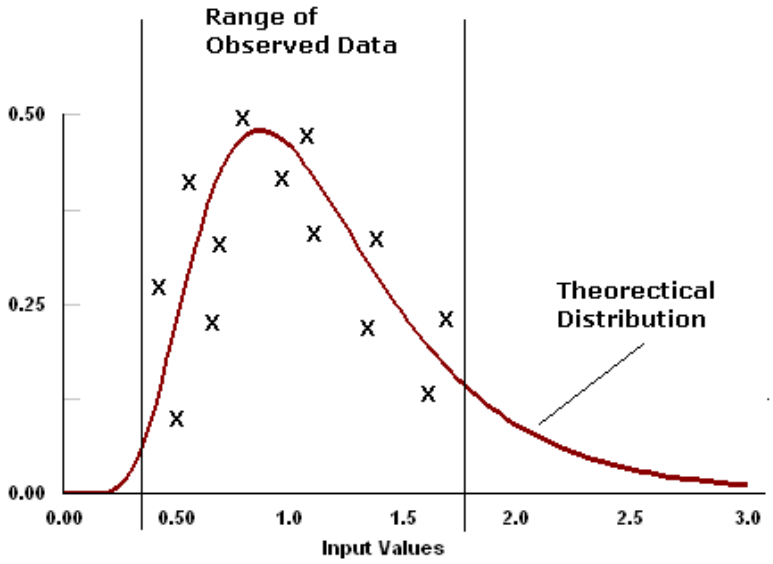


Figure 4.12 Theoretical Distribution Exercises Limits of Data

When an input variable is approximated with a probability distribution, the simulation analyst must be certain that the distribution does in fact represent the variable accurately. The first step in this process is to determine the specific form of the representative distribution. Examples of distributions include exponential, lognormal, normal and Poisson (see Figure 4.13).

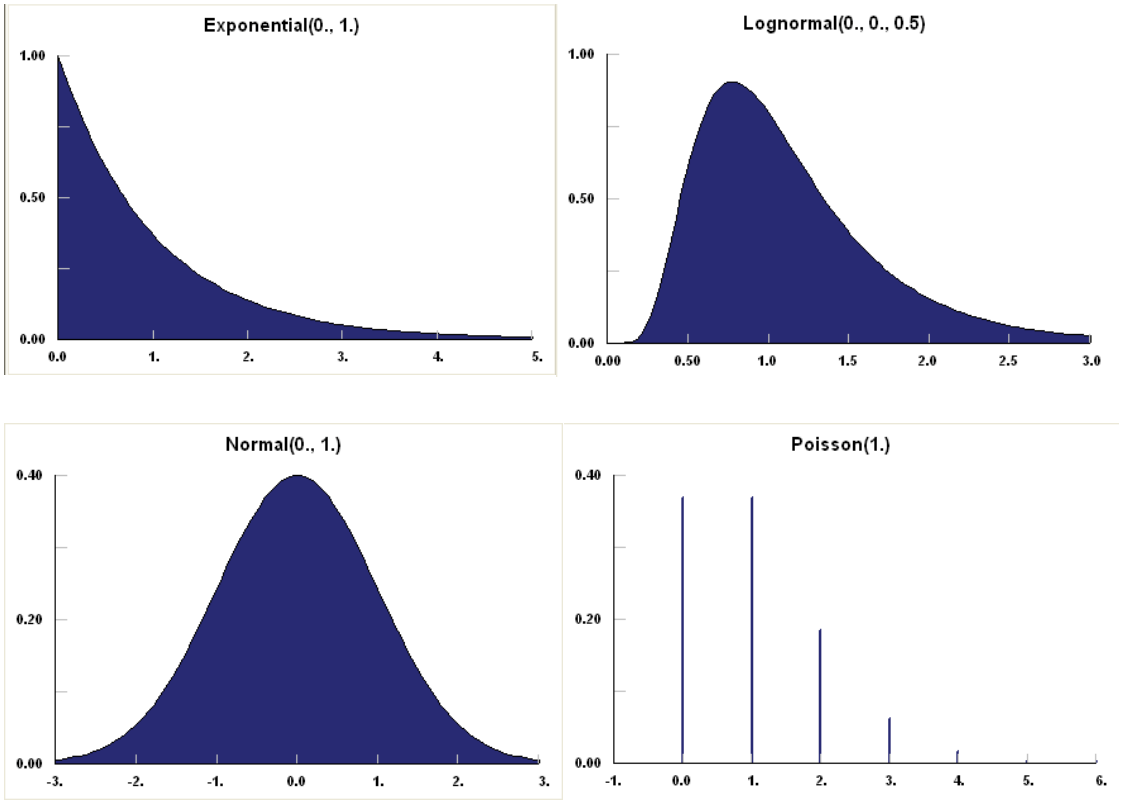


Figure 4.13 Example Distributions

In order to choose the proper distribution, the statistical properties of the collected data must be determined. Initially, this can be done with a visual inspection of the data in histogram form. To confirm the visual inspection, a goodness-of-fit test can be used to measure how closely the observed values match expected values. Two common methods of confirmation are Pearson's chi-square test and the Kolmogorov-Smirnov test. An example of performing a goodness-of-fit test will be provided in sections 4.3.9 and 4.3.10 but simulation analysts rarely perform goodness-of-fit tests manually. Most analysts indirectly apply these tests using features built into statistical analysis software packages that aid in distribution determination. Section 4.3.11 provides details on a more practical method for performing goodness-of-fit testing.

4.3.8 Pearson's Chi-Square Test

Pearson's chi-square test can be used to assess goodness-of-fit and determine whether an observed frequency distribution is significantly different from a theoretical one. In a general sense, the chi-square test relies on calculating the chi-square statistic (X2) which represents the sum of the squared differences between observed data values and theoretical frequency for each possible outcome. Additionally, degrees of freedom, or number of variables that can vary freely, also must be calculated. The value of the test-statistic becomes:

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Where:

X^2 is the test statistic that asymptotically approaches a true χ2 distribution.
O_i is an observed frequency;
E_i is a theoretical frequency;
n is the number of possible outcomes of each event.

The chi-square statistic is used to calculate a p-value derived from chi-square distribution. In general, the chi-square goodness-of-fit test should be applied to a categorical variable from a single population to determine if the sample data are statistically indistinguishable from a hypothesized or theoretical distribution. The following conditions or assumptions must be true for the chi-square test to be valid:

- A simple random sampling method was used to collect the new data.
- The population is 10 times or more greater than the sample.
- A categorical variable is being tested.
- Each level of the variable is five or more.

Then, the following steps are used to conduct the goodness-of-fit test: (1) formulate hypotheses, (2) state analysis parameters, (3) test sample data, and (4) interpret and report finding.

Formulate Hypotheses

In most goodness-of-fit tests, the null hypotheses will be stated to indicate the nature of the test. For instance, the test is seeking to support the premise that the sample data are consistent with a particular theoretical population distribution. For goodness-of-fit testing, a generic null hypothesis becomes:

H0: The collected data are consistent with the theoretical distribution.

Analysis Parameters

Analysis parameters include the rules for accepting or rejecting the null hypothesis. In general, the parameters include significance level and test method.

Test Sample Data

In order to test the sample data against the theoretical population, degrees of freedom, expected frequency counts, test statistic, and p-value must be determined.

- Degrees of freedom (DF) equals one subtracted from categorical variable's number of levels (k):
  $DF = k - 1$
- Expected frequency for each level of the categorical variable equal sample size multiplied by the hypothesized proportion of the theoretical distribution:
  $Ei = npi$
- The chi-square test statistic is a random variable defined as:
  $X2 = \Sigma [(Oi - Ei)2 / Ei]$
- The p-value represents the probability of observing a sample statistic as extreme as the test statistic.

Interpret Results

A conclusion is developed by comparing the p-value to the significance level. If the p-value is less than the significance level then the null hypothesis is rejected.

4.3.9 Example Chi-Square Goodness-of-Fit

The following scenario illustrates the chi-square goodness-of-fit test.

Chi-Square Goodness-of-Fit Example

A toll booth simulation is being updated. Originally, different booth configurations had been tested using a stream of vehicles expected to pass through the booths. Data representing the traffic had been collected for the original model when it was constructed five years earlier. The study used a distribution of vehicle types based on highway observations. At that time it found that approximately 10% of the vehicles passing through the booth would be motorcycles, 60% cars, 20% trucks and 10% cars with trailers. Since that time, the demographics of the surrounding area had changed dramatically and traffic patterns had shifted to a mix that represented industrial growth in the region. A small random sample of data was collected to determine if the original distribution was still representative of current conditions. In order to make this determination, the new random sample was tested against the original theoretical distribution using a chi-square goodness-of-fit test. This would provide statistical evidence that the new sample distribution either differed significantly from the original distribution or that it was essentially the same.

An analysis was performed on the following sample data (100 observations):

Motorcycles: 5%
Cars: 55%
Trucks: 35%
Cars with Trailers: 5%

A visual inspection of the data is provided in Figure 4.14.

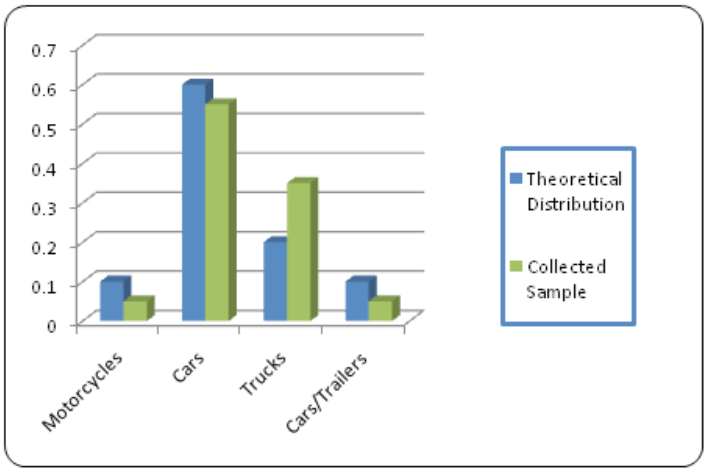


Figure 4.14 Visual Comparison between Collected Sample and Theoretical Distribution

While the distributions are similar, a noticeable difference exists that indicates a statistical test should be completed.

Chi-Square Goodness-of-Fit Example (con't)

Hypothesis:

H_0: The collected data are consistent with the theoretical distribution of 10% motorcycles, 60% cars, 20% trucks and 10% cars with trailers.

Analysis parameters:

In this analysis, a .05 significance level will be used which means the results provide a 95% confidence interval regarding the conclusion. The test method has already been stated and is the chi-square goodness-of-fit test.

Test sample data:

The chi-square goodness-of-fit test statistic and p-value for the sample data are computed as follows:

Degrees of Freedom:

DF = k - 1 = 4 - 1 = 3

Expected Frequencies:

$(E_i) = n * p_i$
$(E_1) = 100 * 0.10 = 10$
$(E_2) = 100 * 0.60 = 60$
$(E_3) = 100 * 0.20 = 20$
$(E_4) = 100 * 0.10 = 10$

Chi-Square Test Statistic:

$X^2 = \Sigma [(O_i - E_i)^2 / E_i]$
$X^2 = [(5 - 10)^2 / 10] + [(55 - 60)^2 / 60] + [(35 - 20)^2 / 20] + [(10 – 5)^2 / 10]$
$X^2 = (25 / 10) + (25 / 60) + (225 / 20) + (25 / 10) = 16.67$

p-value:

The p-value can be obtained through use of a distribution table shown in Table 4.6:

From the table, $p < 0.001$

df\area	0.9	0.5	0.25	0.1	0.05	0.025	0.01	0.005	0.001
1	0.02	0.45	1.32	2.71	3.84	5.02	6.63	7.88	10.83
2	0.21	1.39	2.77	4.61	5.99	7.38	9.21	10.60	13.82
3	0.58	2.37	4.11	6.25	7.81	9.35	11.34	12.84	16.27
4	1.06	3.36	5.39	7.78	9.49	11.14	13.28	14.86	18.74
5	1.61	4.35	6.63	9.24	11.07	12.83	15.09	16.75	20.50

Table 4.6 Values of X2 for Various Probabilities

A more common approach for goodness-of-fit testing is to use a distribution calculator or statistical analysis program rather than a table. Various distribution calculators can be located on the Internet either as commercial products or as freeware (see Figure 4.15). The calculator illustrated below was developed by Hans Lohninger at the Institute of Chemical Technologies and Analytics, Vienna University of Technology and can be obtained from:

http://www.vias.org/simulations/simusoft_distcalc.html

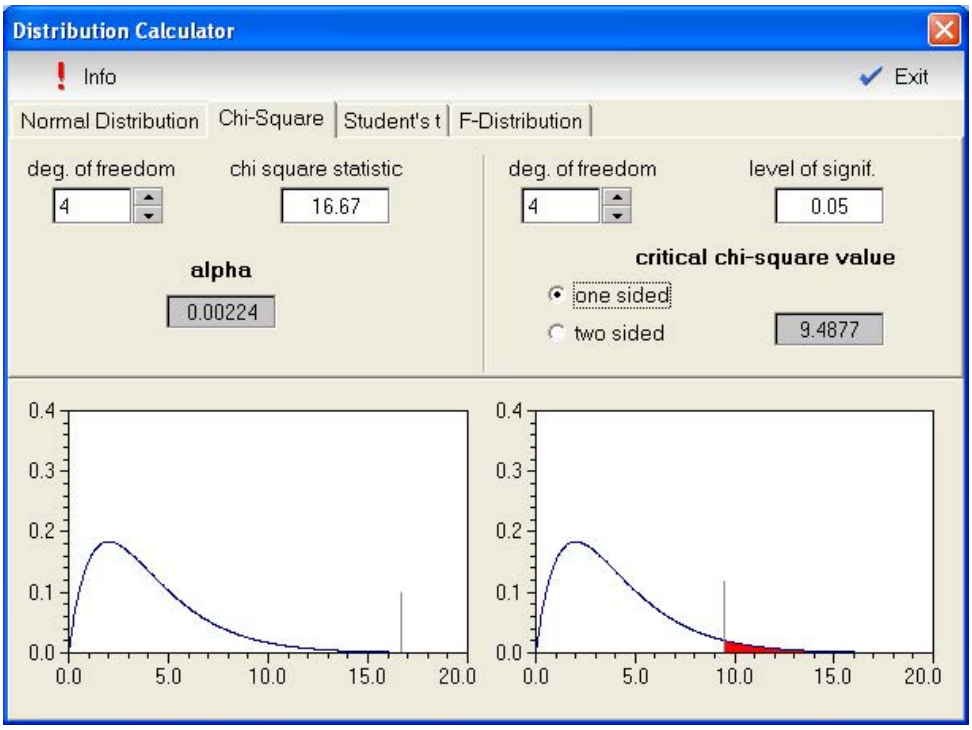


Figure 4.15 Distribution Calculator

As can be seen from both the chart and the Distribution Calendar, the p-value (.00224) is less than the significance level (0.05), therefore the null hypothesis cannot be accepted. The original data distribution cannot be used in the new simulation. A larger data sample should be collected and the input distribution must be redeveloped.

4.3.10 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is another accepted nonparametric method of verifying goodness-of-fit between distribution function of the collected data and a theoretical probability distribution function. It has several advantages over the chi-square test:

1. Does not require data to be grouped in any way, thus keeping data from being lost or inappropriately classified.
2. No interval specification or class breakdown is necessary.

Kolmogorov-Smirnov also has several disadvantages:

1. The data being tested must be continuous. Discrete data should not be used (although it sometimes is used, particularly when estimates are being developed).
2. Parameters for the population cannot be estimated from the sample's characteristics as they are in the chi-square method.
3. It is more sensitive near the center of the distribution and less sensitive in the tails.

4.3.11 Anderson Darling Test

The Anderson-Darling test determines whether a sample of collected data comes from a hypothesized distribution. It is closely related to the Kolmogorov-Smirnov test but gives more importance to the tails. Unlike the K-S test, the Anderson Darling test uses the hypothesized distribution to calculate critical values which makes the test more sensitive.

4.3.12 Example Using Data Fit Software

Most simulation analysts use distribution fitting software as a means of determining suitable input distributions for their models. Two commonly used software packages for doing this are:

ExpertFit from Averill Law & Associates:
http://www.averill-law.com/ExpertFit-distribution-fitting-software.htm

StatFit from Geer Mountain Software Corporation:
http://www.geerms.com/

Student Union Banking Center

A Midwestern university in the United States considered opening a banking center next to its busy student union food court. As part of the cost-benefit analysis, a computer model was constructed. To accurately simulate staffing requirements, data was collected from a bank at a similar university. Data was obtained through direct observation. The analysis team clocked the arrival times of each individual as they entered the banking area, the length of time they interacted with the teller(s), and their total time in the system (which included queuing time). Approximately 50 observations were collected. In order to use the service time data in the model, Stat::Fit Version 2 from Geer Mountain Software was used to compare the collected data to over 30 known theoretical distributions and test for goodness-of-fit. Nonparametric tests were used since the collected data was not normally distributed. Stat::Fit Version 2 tests the following null hypothesis:

H_0: Fit of collected data to theoretical distribution is good.

The best fitting distribution for the service time data was lognormal. This hypothesized distribution was tested against the data using both Kolmogorov-Smirnov and Anderson Darling tests. In both cases, the null hypothesis H_0 could not be rejected and an acceptable fit was indicated. Figure 4.16 looks at the hypotheses tests. Figure 4.17 provides a graphic look at the collected and hypothesized data. Figure 4.18 provides additional statistical output.

The following example demonstrates how this software is used:

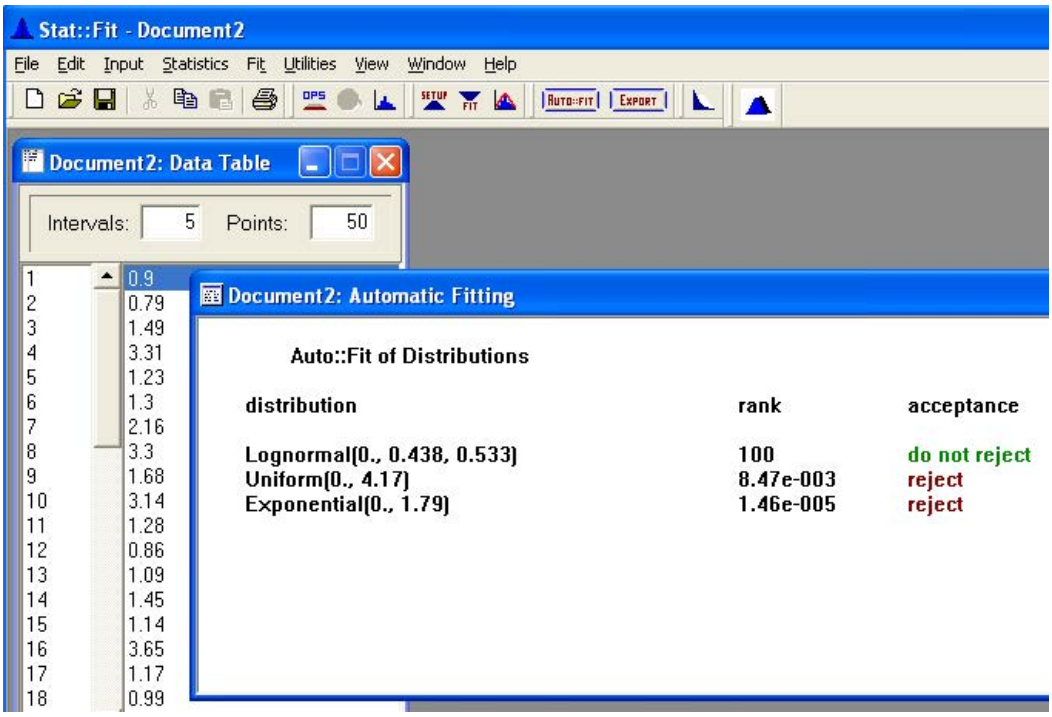


Figure 4.16 Hypothesis Tests

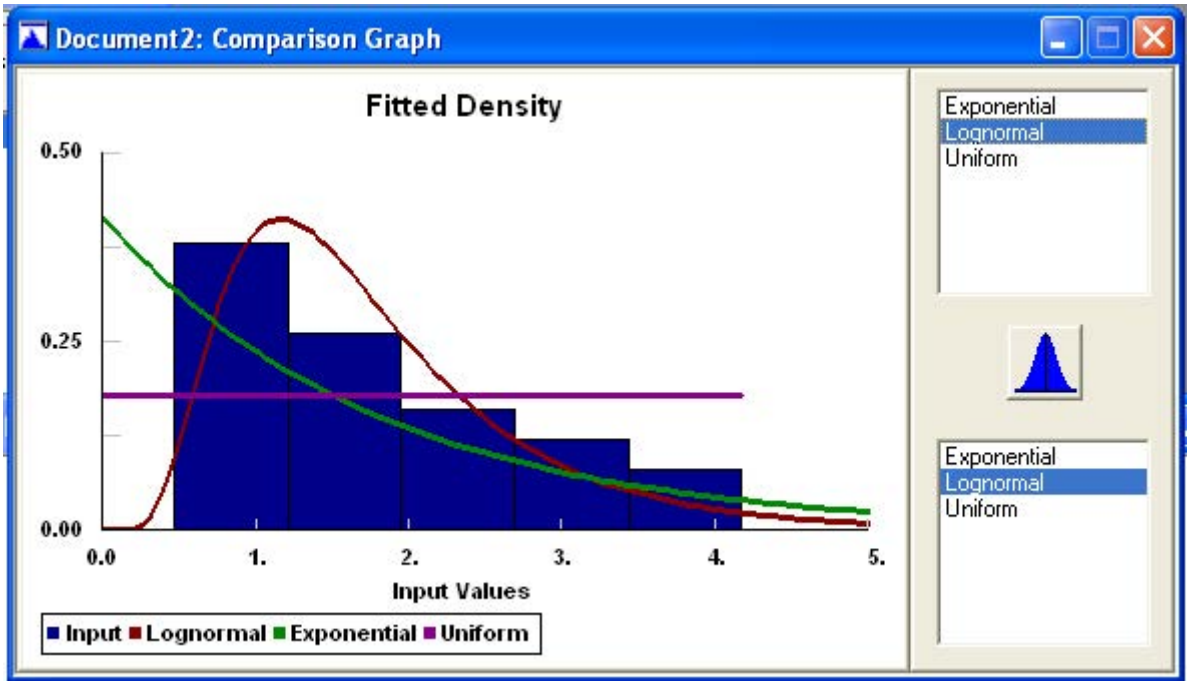


Figure 4.17 Comparison between Collected and Hypothesized Data

Comparison between Collected and Hypothesized Data

Figure 4.18 Statistical Output from Stat::Fit 2

Distribution fitting software packages such as Stat::Fit 2 often provide additional features to translate the statistical distribution into code that can be inserted into a simulation model directly. Figure 4.19 provides the code required to use the lognormal distribution tested above in GPSS.

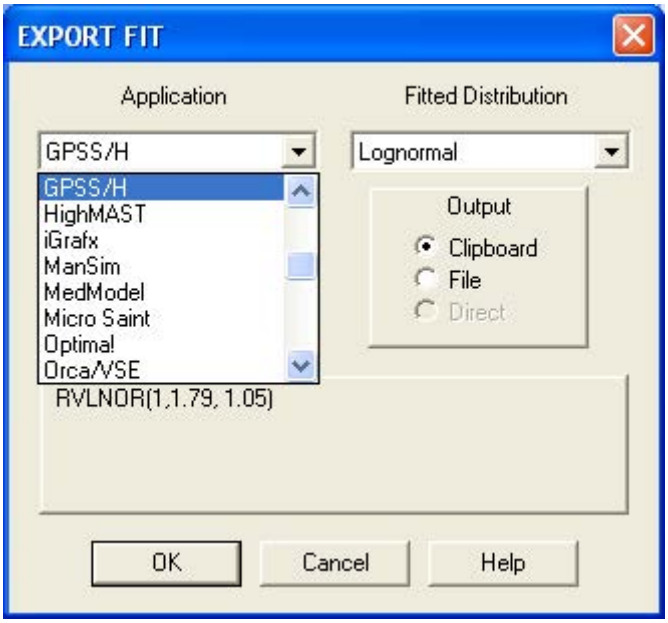


Figure 4.19 GPSS Code for Lognormal Distribution

4.4 Human Component Considerations

Special consideration often must be made when human behaviors are included in models. According to Brown (2002) variability due to the human factor can be attributed to at least three sources:

1. Different people may have different backgrounds and base skill levels.
2. Different people may have different levels of experience with both the system and the benchmark tasks.
3. A level of inherent variability exists in all human behavior: two people with the same training and background may still behave differently for unknown reasons.

To mitigate and manage this variability, different techniques can be used. Among these are Simplementation Cases and Persims (see McHaney 2008 for a complete discussion on this topic).

4.4.1 Simplementation Cases

A use case facilitates the capture of system functional requirements during systems analysis and design in traditional software development. In a use case potential users, known as actors, are documented along with their reason for using the system. This way, information captured with a use case ensures the developed system accounts for all user needs.

Use cases are part of the Unified Modeling Language (UML) and help transform unstructured user descriptions of a desired system into a more formal model. Through a conceptual extension, a use case methodology can be applied to the human element in a simulation model. This will help develop a better understanding of how that system will be impacted by the human variability and other aspects of the human component. Instead of showing how humans will use the system, the simplementation case will represent how a human acts in the system. The actor is moved from interacting with the system to becoming a component within the system. Documenting simulation actors is important for several reasons:

1. Model developers are better able to see the full scope of human action within the system.
2. The places where human behavior will impact the system are more closely examined. For example, a bank teller could be represented with a single actor on the diagram but a range of personality types and behavior patterns might be required to realistically capture all the individuals that might play that role in the real world.
3. The analyst is forced to consider scaling issues in model development.

Whenever possible, each actor should be described in the context of a single simplementation case. This is not a firm rule and occasionally it might make more sense to show multiple actors in a single case when the diagrams are not complicated or difficult to understand.

The sum total of the simplementation cases ensures the modeled system considers all human behavior within the model. Further, simplementation cases become a good mechanism to organize unstructured system descriptions into formal qualitative model requirements. The actors become system components with potential for complex behaviors. Figure 4.20 illustrates.

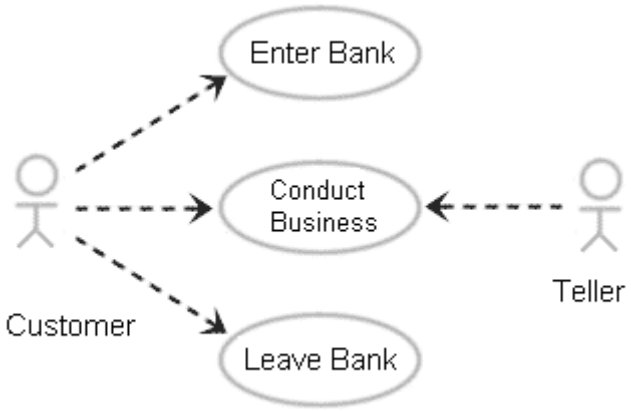


Figure 4.20 Simple Simplementation Case Example for Bank Customer and Teller

4.4.2 Persims

While general roles for the human component will become apparent during simplementation case construction, the range of behaviors for these actors will not be fully documented. For this reason, and particularly in systems where the human component may have a major impact, the primary actors should be extended through the development of a construct equivalent to personas in a software development project.

In software development, personas are fictionalized descriptions of system user groups or archetypes. These archetypes are broken down by behaviors, goals, and motivations to extend use cases beyond mere function and create realistic profiles of potential users to give system developers a better sense of the system. Personas help designers overcome issues related to narrow thinking and provide contradictory views of intended groups of users.

When personas are extended for use in simulation models, they become Persims, which stands for persons that are simulated. Persims extend the understanding of the actors represented in general simplementation cases and help analysts realistically scale models and better consider ranges of behavior for the human component in models.

Generally, one actor in a simplementation case will represent several persims, each with their own experience and behavioral motivations within the system. If various persims can be identified and studied, then they may be represented with different input distributions and hence model scaling will become more refined and result in a more accurate simulation.

Potential persims can be obtained in a variety of ways. Among these are (Table 4.7):

- In an existing system, human behaviors can be observed and codified into archetypes.
- Behaviors can be developed from interviews, transcripts and surveys.
- Theoretical archetypes can be developed for new systems being simulated.

Table 4.7 Ways to Obtain Persims

For instance, archetypes for tellers at the university student union food court bank might be:

1. students needing part time employment
2. spouses of professors
3. tellers from the main branch seeking a different experience

In a simplementation case, these archetypes would be represented by a single actor but when considered in the light of their motivations and experience, each persim might behavior within the system according to a different distribution and this scaling may have a big impact on system outcomes. Table 4.8 provides an example of the three bank persims and provides a starting point for collecting data to model each.

Persim Brief Summary					
Name	Archetype	Description	Personal Goals	System Interaction Notes	Representative Portion of Group
Gabe Capman	Senior Teller	Knowledgeable, long-term employee, quick problem solver, sometimes helps other teller at the expense of his own customer's time, enjoys coffee breaks	To continue to increase knowledge and appear competent at all times, be helpful and available for answering questions, advance to a managerial position	Fast and accurate, can speed up when customer queues lengthen	20%
Neve Irenite	Young Teller	Enthusiastic, takes extra time to be friendly to customer, makes occasional mistakes, is very careful and checks work frequently	Learn the system; make fewer mistakes; be a customer pleaser	Cannot speed up when lines build, gets flustered and more mistake prone when time becomes an issue, prone to explain everything carefully to customers	70%
Bob Woffer	Unhappy Teller	Disgruntled and has plans to quit, generally polite to customers but not interested in much personal banter, avoids trying to help others, works slowly	Would prefer to enter own information regarding seminars, Wants to do things fast and not worry about transaction that didn't go through properly	Not willing to help others and often slips out when queues build	10%

Table Adapted from McHaney, 2008.

Table 4.8 Persims Identified in Midwestern University Food Court Bank

4.5 Bibliography

M.L. Berenson, D.M. Levine and M. Goldstein, Intermediate Statistical Methods and Applications: A Computer Package Approach." Prentice-Hall, Inc. Englewood Cliffs, New Jersey (1983).

S. Calde, K. Goodwin and R. Reimann, SHS Orcas: The first integrated information system for long-term healthcare facility management. Conference on Human Factors and Computing Systems, Case studies of the CHI2002/AIGA Experience Design Forum, ACM Press, New York (2002).

A. Cockburn, Writing Effective Use Cases, Addison-Wesley, Reading, MA (2000).

A. Cooper, The Inmates are Running the Asylum. SAMS, Indianapolis, IN (1999).

Distribution Calculator Information, Hans Lohninger Website, Inst. of Chemical Technologies and Analytics, Vienna University of Technology, http://www.vias.org/simulations/simusoft_distcalc.html, Retrieved July 8 (2009).

ExpertFit Information, Averill Law & Associates, http://www.averill-law.com/ExpertFit-distribution-fitting-software.htm, Retrieved July 11 (2009).

M. Fowler, UML Distilled: a Brief Guide to the Standard Object Modeling Language, Addison-Wesley, Reading, MA (2004).

F.N. Kerlinger, Foundations of Behavioral Research, Harcourt, Brace, Jovanovich College Publishers, Fort Worth, Texas (1986).

J. Kleijnen, Statistical Tools for Simulation Practitioners. Marcel Drekker, New York (1987).

J.J. Komo and W.J. Park, "Decimal Pseudo-Random Number Generator," Simulation (57:4), 228–230 (1991).

A.M. Law and W.D. Kelton, Simulation Modeling and Analysis, 3rd Edition, McGraw-Hill Book Company (2000).

A.M. Law and C.S. Larmey, An Introduction to Simulation Using SIMSCRIPT II.5. CACI, Los Angeles, California (1984).

R.W. McHaney, Computer Simulation: A Practical Perspective, Academic Press, San Diego (1991).

R.W. McHaney, Simulation, In Miriam Drake (Ed.), Encyclopedia of Library and Information Science, Second Edition, Marcel Dekker, New York, 2643–2655 (2003).

R.W. McHaney, "Use-Cases and Personas: Uses in Service Sector Simulation Development," International Journal of Simulation and Process Modelling (4:3–4), 264–279 (2008).

S.P. Murphy and T. Perera, "Successes and Failures in UK/US Development of Simulation," Simulation Practice and Theory 9(6-8), 333–348 (2002).

ORMM Information, Paul Jensen's Website, http://www.me.utexas.edu/~jensen/ORMM/index.html, Retrieved July 8 (2009).

E.J. Pedhazur and L.P. Schmelkin, Measurement, Design, and Analysis: An Integrated Approach, Erlbaum, Hillsdale, NJ (1991).

J. Pruitt and T. Adlin, The Persona Lifecycle: Keeping People in Mind Throughout Product Design, Morgan Kaufman, Amsterdam (2006).

J. Pruitt and J. Grudin, Personas: Practice and Theory. Paper presented at Designing for User Experience, San Francisco, CA (2003), Accessed online (June 2006) at: http://research.microsoft.com/copyright/accept.asp?path=http://www.research.microsoft.com/research/coet/Grudin/Personas/Pruitt-Grudin.doc&pub=ACM.

Queuing Analysis ToolPak 4.0 Information, http://www.business.ualberta.ca/aingolfsson/QTP/default.htm, Retrieved July 7 (2009).

G. Randolph, "Use-Cases and Personas: A Case Study in Light-Weight User Interaction Design for Small Development Projects," Informing Science (7), 105–116 (2004).

StatFit Information, Geer Mountain Software Corporation, http://www.geerms.com/, Retrieved July 11, 2009.

D. White and R.W. McHaney, Simulation Steps, Working Paper (2009).

H. Woo and Robinson, W. "A Light-Weight Approach to the Reuse of Use-Cases Specifications," In Proceedings of the 5th annual conference of the Southern Association for Information Systems. Savannah: SAIS, 330–336 (2002).

5 Simulation Quality and Development

Building the model is often the step in the simulation life cycle analysts anticipate eagerly. The process begins with selecting a tool or language for development and ends with a complete model, ready for experimentation and output analysis.

5.1 Quality Assurance Phase

Quality assurance is an ongoing activity in simulation model development. That being said, several activities are keys to ensuring a high quality simulation emerges during development. Among these are validation concerns.

5.1.1 Validation

Validation is used to determine the real-world system being studied is accurately represented by the simulation model. In other words, this process ensures the conceptual model is correct and establishes an acceptable level of confidence the conclusions drawn from running the simulation will give insight as to the true operating characteristics of the system being modeled. This confidence, called face validity, should radiate from both modeler and model user.

Validation is ongoing and should be part of the simulation process from the very beginning and continue until the very end. Validation techniques are particularly important during the phases of the simulation process where input data are collected, analyzed and structured for use in the model.

5.1.2 Simulation Input Data Validation

As stated in an earlier section, simulation inputs are gathered in two formats, qualitative and quantitative. Qualitative inputs are the underlying assumptions, rules, and other non-numeric data that will be used in the simulation. This information should be gathered and validated through one or several of the following methods:

1) Observation: If a model of an existing system is being developed, the analyst can observe different situations and ensure that the assumptions to be used in the model are valid.

2) Expert's Opinions: A list of assumptions and rules can be evaluated by experts. A modeler should interact with both system experts and model users throughout the life of a simulation project beginning with development of the input data.

3) Intuition and Experience: If a simulation analyst frequently models systems that share many common characteristics, he/she will develop an intuitive feeling that will help give the model added validity.

Quantitative or numeric inputs for a simulation should be tested for validity in the following ways:

1) Statistical Testing: If a theoretical input data distribution is being used to model empirical data, chi-square, Kolmogorov-Smirnov, or other goodness-of-fit tests should be used to assess that data. If the fit is close, the theoretical data can reliably be used as a valid representation.

2) Sensitivity Analysis: Involves altering the model's input by a small amount and checking the corresponding effect on the model's output. If the output varies widely for a small change in an input parameter, then that input parameter may need to be re-evaluated. Highly sensitive model behavior may indicate problems with the coding or input data.

5.1.3 Validation of Simulation Outputs

Often, the best test of model validity will not come until later in the simulation life cycle. Most of the time, the best test is to conduct a thorough analysis of the simulation's output data. If the model's output data closely represents the expected values for the system's real-world data, then validity is more likely.

When a model has been developed for an existing system, a validity test becomes a statistical comparison. Data collected from actual system operation can be used as a theoretical comparator. When the system does not yet exist, validity becomes harder to prove. In many cases, validity cannot be definitely proven until some point in the future when the system being modeled has been installed and is operational. Before that time, several methods can be used to increase the confidence level that a model is valid:

1. Comparison with data from a similar system: If a system exists which is similar in nature to the one which is being modeled, an interpolation of system output can be derived and compared to the simulation.
2. Expert opinions: An expert on the type of system being modeled can be consulted and shown the output data. His/her opinion will help lend more confidence that the model is valid.
3. Calculated expectations: In some cases, expected output can be calculated and the model output compared to the result. If the model is too complex, it may be possible to analyze individual subsystems and do calculations on each part to help establish validity.

5.1.4 Summary of Validation

Validity is an important aspect of simulation. Without confidence that a simulation is valid, model output is worthless. One key to validity is communication. With continual interaction between system designers, component vendors, experts, model users and system analysts, the perceived and actual validity of a model will be greater.

5.2 Selection of a Language or Tool

Following completion of a concept model, a decision must be made to halt the effort and be content with currently available information or proceed with development of a detailed simulation. If further development is the chosen course of action, then an appropriate simulation language or tool must be made.

As mentioned in an earlier chapter, simulation software products can be broken into two major categories: simulators and simulation languages. Simulators are application specific prewritten modeling tools. Simulation languages, on the other hand, are general purpose programming languages specially adapted for modeling applications.

Choosing a software product can be a difficult process. Software is an intangible entity with numerous parameters and features to compare. Many vendors, based on their own testing criteria and definition, claim to have the fastest, most user-friendly, or best package available. While it may not be possible to select an "absolute best" simulation package, it is very possible to select a simulation package that will meet the requirements of a particular project.

Software evaluation can be broken into four steps (Figure 5.1):

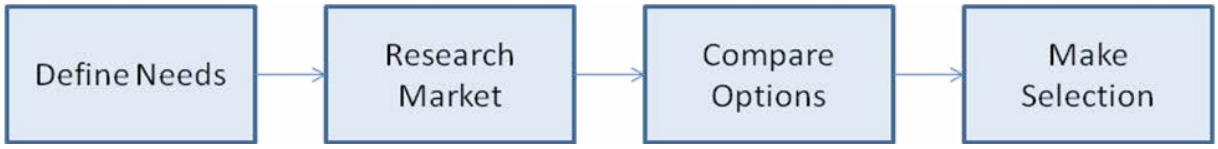


Figure 5.1 Software Evaluation Activities

Step one: Define needs – This is an information gathering step during which the prospective software purchaser needs to answer several questions.

1. Determine the frequency of future simulation work. Will this be a onetime project or will simulation be used regularly from this point on?

   If a simulation is to be done only once for this project, the amount of time spent on the learning curve should be minimized. This can be accomplished by purchasing a user-friendly simulator or by hiring a simulation consultant. If simulation work is to be done on a regular basis in the future, then serious development of an in-house simulation capability is desirable. This development would involve the training of appropriate personnel and the selection of either a general purpose simulation language or appropriate simulator.

2. What type of system will be modeled? Is it unique in function?

   If the system to be simulated is unique, then it will be hard to find a simulator package that can be used without trouble. In this case a general purpose simulation language should be used. If the system being modeled is standard materials handling equipment being purchased for in-house use, the probability that a simulator is available is quite high.

3. Who will be doing the simulation work?

   If an engineer who is not a programmer and not a computer expert will be doing the work, a simulator package may be desirable. On the other hand, a simulation analyst might feel constrained by a simulator package and might desire the full range of capabilities a language offers.

4. What budgetary constraints exist?

   The purchase of all tools and equipment is dependent upon a budget. Simulation software products exist in nearly all price ranges. Budget will have an impact on what can be bought.

5. Who will be using the results?

   Will this simulation study be used to "sell" an idea to management? How impressive does the output need to be? If it is going to be used as a research tool, then good statistical output is desirable. If it will be used to impress a potential customer or upper management, then a top notch animation might be desirable.

Answers to the preceding five questions will help start the software evaluation process by giving definition to the general needs of the prospective buyer.

Step Two: Research Market – A large and varied simulation software market exists. Based on the needs analysis conducted in Step One, a list of important software features can be generated. Products exhibiting some or all of these features can then be researched further.

Many simulation software vendors advertise their products in trade magazines popular in a particular target market. Others provide extensive websites with examples of use, free downloads of trial software versions, and successful case studies.

Another source of information about simulation software is the Winter Simulation Conference. Its Website is http://wintersim.org/.

Step Three: Compare Available Options – The process of selecting simulation software is much like shopping for any other commodity. Certain features are more important than others. No one product will be able to provide the best of each desired attribute, so a somewhat subjective decision will have to be made.

In order to facilitate the process of comparing different software packages, a list of important attributes needs to be generated. Each attribute in this list should be given a weighting factor. In this way an evaluation template is created. An evaluation template can be easily creating using spreadsheet software. The following example illustrates (Table 5.1):

	Weighting Factor		Product Score		Total
Graphics	3	X	_____	=	_____
User-Friendliness	8	X	_____	=	_____
Software capability	5	X	_____	=	_____
Ease of Use	9	X	_____	=	_____
Power	4	X	_____	=	_____
Cost	10	X	_____	=	_____
Output reports	6	X	_____	=	_____
Ease of Debugging	8	X	_____	=	_____
Statistics	7	X	_____	=	_____
Customer Support	8	X	_____	=	_____
Training	8	X	_____	=	_____
Documentation	8	X	_____	=	_____
Vendor stability	2	X	_____	=	_____
Application specific modules	9	X	_____	=	_____
			Total Product Score =		_____

Table 5.1 Simulation Software Evaluation Template

T-Co's Search for Simulation Software

T-Co is a manufacturer of t-brackets, hinges, and other small hardware items. They plan to install a new series of CNC machines to increase their capacity. Before making a major investment, management at T-Co decided to simulate the proposed system. They identified the following attributes as being important in the simulation software selection process:

- This will be a onetime simulation project.
- The system being modeled is not unusual or unique.
- An in-house engineer will be doing the simulation work.
- An inexpensive package should used.
- Results will be for in-house analysis. Animation is not important, but would be nice.

The engineer assigned to the simulation project at T-Co developed the evaluation template in Table 5.1 to help in the software selection process. The weighting factors (1 to 10, with 10 being most desirable) were subjectively developed from the attributes identified by management. The weighting factor is multiplied by the product score to produce a column of totals. These totals are added to give an overall numeric product score.
Before the engineer was able to really get started on his evaluation process, the management at T-Co decided simulation could be used for other future in-house projects. They went through their requirement list and altered it to read as follows:

- Simulation will be used on a regular basis at T-Co.
- The systems being modeled are not unusual or unique.
- An in-house engineer will be doing the simulation work.
- The best package, without regard to cost, should be used.
- Results will be used both for in-house analysis and as a tool to sell our ideas to our parent company. Animation can be used to help perform this function.

The project's engineer received the new requirement list and modified his template as shown in Table 5.2.

Table 5.2 Modified Simulation Software Evaluation Template

	Weighting Factor		Product Score	Total
Graphics	10	X	_____	= _____
User-Friendliness	6	X	_____	= _____
Software capability	10	X	_____	= _____
Ease of Use	5	X	_____	= _____
Power	9	X	_____	= _____
Cost	2	X		=

Table 5.2 Modified Simulation Software Evaluation Template

Step 4: Make Selection – When all the options have been carefully researched and a template has been used to develop a product score, it is time to make a final selection.

Most software vendors offer either training seminars, trial use periods, or have limited capability versions of their software available for download on their websites. Rather than purchasing the software package with the highest product score, several top contenders should be investigated more closely. In this way intangible features of the software can be subjectively judged firsthand. Then the package best suiting the goals of the simulation project can be selected for usage.

5.3 Model Construction

Models may be constructed using a simulation language, simulator, or simulation software development environment. The construction stage of a simulation involves writing or constructing computer code to accurately represent the system being modeled. Prior to coding, a flow chart or block diagram should be developed as a means of thinking through the model. In addition to being a design tool, flow charts and block diagrams provide formal documentation illustrating a model's logic.

The process of writing the model's code will be somewhat dependent on the simulator or simulation language chosen. However, by this time in a simulation project, the system has a preliminary design, the model has been thought out and conceptually designed and input data has been analyzed. This reduces the actual construction of a model to an almost clerical function. Some creativity must be used to find an appropriate means of representing an entity or process in terms of the constraints imposed by the software being used, but otherwise, model construction is a relatively routine procedure.

Careful Documentation: One of the best forms of verification is the careful documentation of a simulation model. This includes both documenting the programmed code and all the assumptions that have gone into building the model. The documentation process forces the programmer to think through his/her logic one more time and put this thought process down in writing.

Structured Programming Approach: A structured approach to programming involves breaking the model into logical program modules which can each be individually tested, debugged, and ensured to work properly. It is much easier to debug twenty modules, each consisting of fifty lines of code than it is to debug a single thousand line program. If the program were run as one large unit, locating errors would be difficult and time consuming.

5.4 Verification

In order to ensure a model will be representative of its real world counterpart, a verification process should be followed. This process extends through-out most of a simulation development project and normally calls for program debugging and manually checking calculations used in the model. It ensures the computer implementation of the conceptual model is correct.

A number of measures can ease the verification task. These can be grouped into two broad categories: preventative verification and appraisal verification.

Preventative verification strives to ensure that the simulation has been created in a way that will minimize errors. Two methods of preventative verification include:

Appraisal verification is done to check or appraise the programming after it has been coded. Several appraisal methods follow:

Trace: Most major simulation languages have incorporated a tool called a trace. A trace enables each programming path to be checked by printing out the values of key variables, counters, and statistics after the occurrence of each event in the simulation. This feature enables the programmer to ensure proper statistical updates and logical decision making are taking place correctly.

Animation: A very popular method of checking program veracity is through the use of animation. When a graphic animation is available, it becomes possible for a programmer to visually verify system operation by watching the computer screen. Although verification in this fashion is easy, it won't guarantee that no subtle problems exist. It is best to use animation in conjunction with other methods of verification.

Run under Simplified Conditions: A good method of verification is to run the simulation using simplified conditions. Hand calculations can be performed to prove that the model operates as expected.

Apply Common Sense to Output Reports: An often overlooked method of verification is the use of common sense. If something in an output report looks unusual or out of place, it probably is. The more experience a simulation analyst has, the better he/she becomes at finding errors in this way.

5.5 Bibliography

A.M. Law and W.D. Kelton, Simulation Modeling and Analysis, 3rd Edition, McGraw-Hill Book Company (2000).

R.W. McHaney, Computer Simulation: A Practical Perspective, Academic Press, San Diego (1991).

R.W. McHaney, Simulation, In Miriam Drake (Ed.), Encyclopedia of Library and Information Science, Second Edition, Marcel Dekker, New York, 2643–2655 (2003).

T.H. Naylor and J.M. Finger, "Verification of Simulation Models," Management Science (14:2), B92 B101 (1967).

F. Neelamkavil. Computer Simulation and Modelling, John Wiley and Sons, New York (1987).

6 Developing a Simulation-Implementation

After the model has been conceptualized, coded, verified, and validated, it is ready to provide information through experimentation. Experimentation is the process of initializing key parameters in the model and setting up production runs to make inferences about the behavior of the system under study. The experimentation process consists of three steps:

Figure 6.1 Experimentation Process

6.1 Experimental Design

The first step in the experimentation process is experimental design. The term experimental design implies that an experiment is being designed or set up. This is done by identifying key parameters and initializing these parameters to values of interest. An example of an input parameter that could be the focus of an experiment is the number of doormen needed at the 21-Club (See section 6.3). The simulation could be initialized with either one or two doormen. Other issues that need to be resolved during the experimental design stage follow:

Length of Simulated Run Time: Simulation run time must be determined during the experimental design stage. Simulations can be either terminating or steady-state. Terminating simulations end at a predetermined time. For example, if a barber shop opens at 9:00 AM and closes at 5:00 PM each day, a terminating model could be set up with run times that end specifically after eight hours of operation. A steady state simulation on the other hand is set up to measure the long run performance of a system. A model running in the steady-state mode is normally run for a period of time and then statistically "reset" to remove any bias caused by unusual start-up conditions. The run is then continued and steady state results are tabulated.

Replications: Since simulation is an approximation, it is important to treat it as such. This means that one run of the model will not usually provide an absolute answer. The random inputs to a simulation will produce outputs exhibiting variability. The model should be run enough times to produce a sample size from which a mean and standard deviation with a satisfactory confidence level can be calculated.

Reseeding Random Number Streams: Each replication should either re-seed the random number streams being used, or start in the steam where the last run left off. If pseudo-random numbers are being used and each run starts with the same seed value, no variability will occur between runs. Each run would be using the same number stream.

Initial Conditions: In some models, it is possible to force the system to a particular state upon start-up. This action may be desirable in the case of a manufacturing system model that will either never naturally cycle to an unusual condition or may cycle to that condition but only after hours of run time. By setting the initial conditions of a model to certain values, experimentation can be facilitated.

Variables of Interest: The analyst will need to know what values are of interest. For instance, are two systems being compared in terms of throughput? Or are the numbers of workstations being determined? Experimentation will need to be constructed considering the values being examined.

6.1.1 Random and Pseudo-Random Number Streams

A variable Z which can assume any value in the range (x,y) with equal probability is defined as being random (Figure 6.7). Random variables can be uniformly or non-uniformly distributed. They may be continuous or discrete. Random variables are selected by chance and are not influenced in any way by past values. Random numbers are widely used in experiments that are dependent upon chance. Examples of these are machine breakdowns, occurrences of rejected parts coming off an assembly line, and service times at an automatic teller machine.

Random numbers can be generated in many different ways. The flip of a coin, drawing numbers from a hat, roulette wheels, and playing cards are examples of physical generation methods. The most common method of obtaining random numbers for use in the computer simulation field is through algebra. A number called a seed is selected and used to produce a sequence of numbers in the following manner (Figure 6.2):

$Z_i = f(Z_i\text{-}1)$

Where Z_i must be:

- Uniformly distributed (Non-uniform distributions can be created from uniform ones)
- Statistically independent
- Non-repeating (for a desired length of time known as the period of the generator)

Figure 6.2 Method for Obtaining Random Numbers for Use in Simulation

Although the resulting number stream from the preceding example is not truly random, if created properly, it will pass most tests for statistical randomness. Number streams generated through algebraic means are known as pseudo-random numbers. The use of pseudo-random number streams has several advantages over the use of true random numbers including:

1. The sequence of numbers is reproducible. This means different versions of a program can be tested using the same input data.
2. The streams can be generated quickly and efficiently for use in a simulation program.

Although it is important to understand the concepts of random number generation and pseudo-random number streams, a simulation analyst will rarely have to create his or her own random number generator. Most simulation software products already have a built in means of creating pseudo-random numbers. As long as care in selecting a seed value (this is usually done for the analyst as well) is taken, very few problems should occur. However, this is not a topic to be taken too lightly. Without a reliable stream of random numbers, simulation results would be rendered invalid.

6.1.2 Special Concerns for Terminating Simulations

Terminating simulations are models that represent a system that starts in a particular state and then terminates or ends after a fixed period of time or when a predetermined condition is reached. Terminating simulations are used to model particular manufacturing operations where the characteristics of the system startup are important to include or in situations where a specific period of time is being analyzed, like weekend ticket sales for a newly released movie. Terminating simulations study a system over a predefined time period like a shift, a day, or a week. An analyst using a terminating simulation makes no attempt to bring the model to a steady state or stable operating condition. Instead, model replications include all startup biases that might exist (as do most real world systems).

It is important to remember that terminating simulations are still dependent on random samples from input distributions and each run must be treated as a single observation. This means replications using different random number seeds must be generated and used to compute a confidence interval or range likely to include the true value of the mean. Since terminating simulation runs produce output data exhibiting statistical independence, the analysis becomes easier.

6.1.3 Special Concerns for Steady State Simulations

Steady state simulations are intended to examine a system from which startup and ending biases are removed. In other words, the analyst seeks to achieve a state in the model where conditions remain stable. While in many instances this might be less realistic than a terminating simulation, it can be useful in determining underlying characteristics of the system and filtering out the statistical noise. Two main challenges exist when developing a steady state simulation. The first is determining whether a steady state condition has been achieved. The second deals with statistical independence of the derived samples.

Several techniques are used to deal with the first challenge. Almost every system experiences a period of time upon startup where resources are being acquired, customers are entering, or the initial stages of the system are in use while subsequent stages have not yet filled. These forces can result in a skewing of output statistics often called initial bias (See Figure 6.3). Several practical techniques are used to avoid the impact of startup bias. These include:

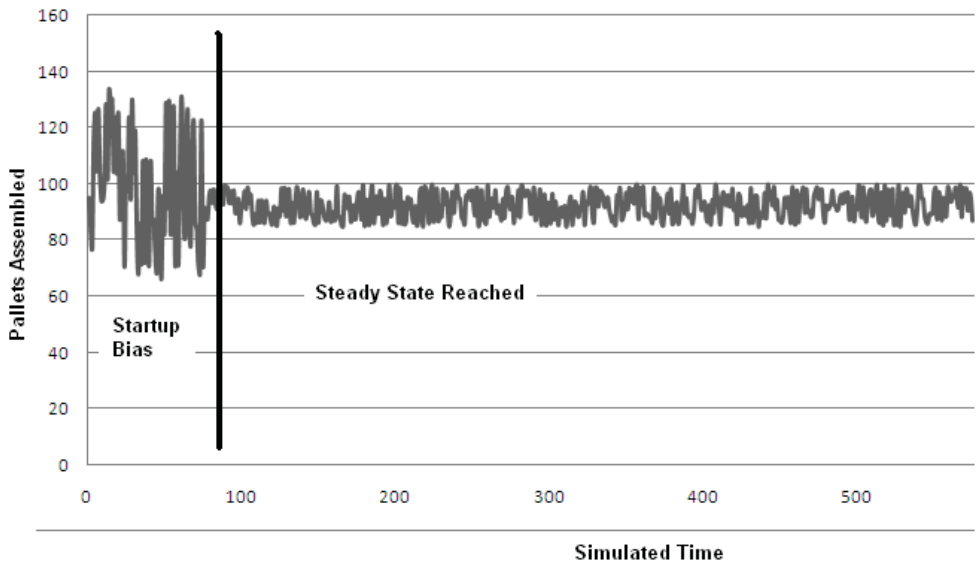


Figure 6.3 Initial Bias in Steady State Simulation

1. A warm-up period is determined after which model statistics collection is restarted. Most simulation software packages have built-in the capability of resetting statistics. In a sense, this would represent throwing out all data collected to the left of the dark line in Figure 6.3 and only considering data after that point in time. The analyst may have to review the simulation results and decide when the model appears to have entered steady state.
2. Initializing the model with a realistic, steady state condition of resource utilization and customers or other entities strategically placed within the model. This technique can work but must be verified as accurate and free of other forms of startup bias. Some researchers call this 'preloading' and others call it priming the model.
3. Data analysis can be used to remove the bias but this technique is rarely used.

The second challenge is to ensure collected samples from the simulation are independent. In a steady state simulation, the current state of the model is dependent on the previous state. Therefore, independence cannot be assumed. In order to correct this problem, a series of simulation runs (often called replications) can be constructed in a way that collection of statistics is suspended between replications while the model runs in steady state. The resources and entities are not cleared, just the statistics. Each replication can be collected with a sufficient run period (almost like the initial bias period) between to ensure independence. Figure 6.4 illustrates. Of course, the analyst would have to ensure the periods selected for statistical collection and those discarded were of sufficient length to ensure the goal of independence is achieved.

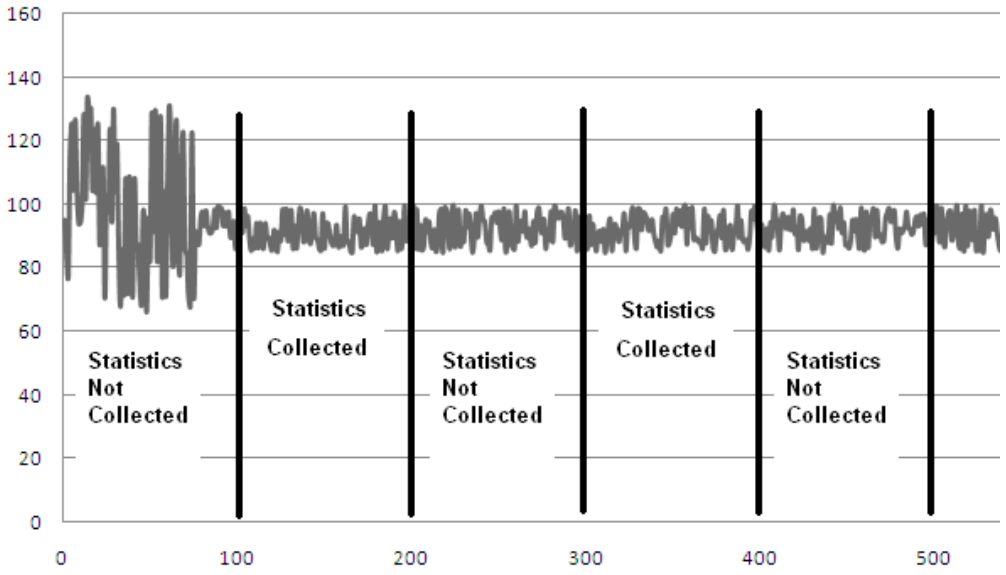


Figure 6.4 Removal of Startup Bias in Steady State Simulation

6.2 Production Runs

After the desired experiments have been identified and the input parameters properly set, production runs can be made. Production runs are no more than executing the simulation code and creating output reports for each desired scenario. Production runs must consider the simulation type (terminating vs. steady state) and the variable(s) to be studied. Enough runs must be completed to construct a confidence interval and ensure collected data is representative of the variation in the system. If a confidence interval is constructed using the output data, the likelihood that the model's true characteristics are included is a probability of 1-α. Three main factors influence the width of a model's confidence interval:

1. Desired confidence level (e.g. α)
2. Replication count – Higher numbers of replications can produce a better (narrower) confidence interval.
3. Variation of parameter being collected (s) – Greater variation produces a less desirable confidence interval and must be offset with more replications.

It is important to note that the collected data represents the model's characteristics. If the model has not been constructed properly, or contains invalid input data or misguided logical assumptions, all the statistical analysis in the world cannot force accurate results to emerge from its operation.

6.3 Output Analysis

The final phase of experimentation involves analysis of model outputs. If the output produced satisfies the simulation project's objectives, the experimentation stage is over. If problems or new questions are raised, the model may be returned to an earlier phase for modification or for production of additional runs.

6.3.1 Statistical Output Analysis of a Single Model

As stated in section 6.2, when a simulation has any type of stochastic behavior incorporated into its structure, it becomes necessary to view the results of a production run as an experimental sample. Enough samples must be taken to ensure that output variability can be accounted for in terms of a mean and standard deviation. Confidence intervals should be created to give the model users a range within which a parameter's true value is likely to fall. The following case study illustrates:

Oslen AGV System

Oslen Manufacturing uses a small AGV system to move material in their warehouse. With a recent increase in orders, they need to coax more productivity out of their aging system. Their goal is to develop an accurate model of the existing system, then later use it to compare various options for improvement. After developing the model, they generated the following set of data representing 100 eight hour shifts. The model uses a terminating approach since it was determined that startup conditions needed to be included for the model to be accurate. The following data was collected (see

44	46	39	52	38
30	45	60	19	35
40	30	40	52	45
65	26	33	45	47
16	32	35	12	29
30	37	14	58	70
47	64	43	37	50
39	60	30	49	23
37	25	48	23	74
49	83	29	18	56
59	49	32	39	38
57	55	51	39	40
14	46	43	61	34
65	19	11	57	31
52	52	55	45	23
31	62	65	54	33
29	56	63	45	29
35	46	57	59	66
59	49	29	33	39
55	54	35	20	10

Table 6.1 Oslen's Data Set

Simulation analysts at Oslen used the following criteria:

1. Desired confidence level (e.g. α) = .05
2. Replication count = 100

In order to determine if the sample size was large enough, Oslen's used SAS software to develop a confidence interval with a 95% chance that the mean for their model's performance had been captured. First they conducted a test for normality to ensure the confidence interval could be constructed with standard statistical practice. SAS provides a variety of statistics for testing normality as shown in Figure 6.5.

Tests for Normality

Test		--Statistic---		-----p Value------
Shapiro-Wilk	W	0.988172	Pr < W	0.5210
Kolmogorov-Smirnov	D	0.055116	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.055786	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.356909	Pr > A-Sq	>0.2500

Figure 6.5 Test Statistics Used to Check Normality

The Shapiro-Wilk W of .988 does not reject the null hypothesis that the variable is normally distributed (p<.521). Likewise, Kolmogorov-Smirnov, Cramer-von Mises, and Anderson-Darling tests do not reject the null hypothesis. Additionally, the plots (shown in Figure 6.6) indicate normality cannot be rejected.

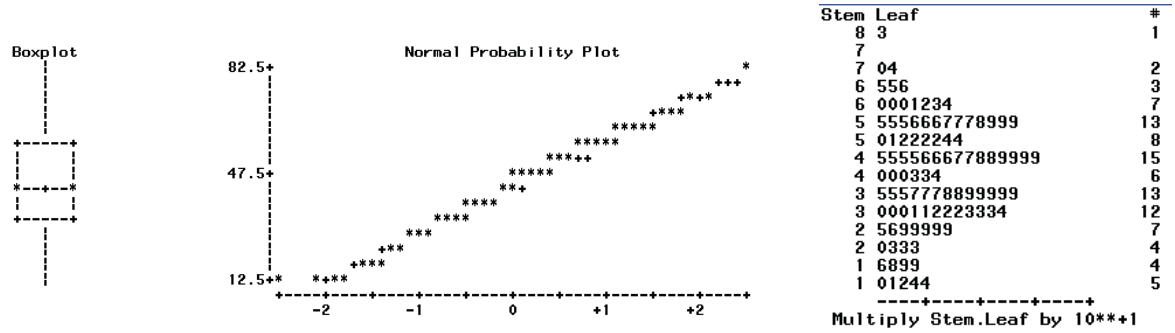


Figure 6.6 Plots Demonstrating Normality of Collected Data

Given that the data is assumed normally distributed, a 95% confidence interval can be constructed to represent the model's mean loads moved per eight hour shift. SAS indicated a mean of 42.49 with a 95% confidence interval (39.45,45.53) as shown in Figure 6.7.

Basic Statistical Measures

Location		Variability	
Mean	42.49000	Std Deviation	15.30927
Median	43.50000	Variance	234.37364
Mode	29.00000	Range	73.00000
		Interquartile Range	23.50000

The SAS System

The MEANS Procedure

Analysis Variable : SIMRUN

Lower 95% CL for Mean	Upper 95% CL for Mean
39.4523094	45.5276906

Figure 6.7 Basic Statistical Measures and Confidence Intervals

6.3.2 Statistical Output Analysis Used to Compare Models

It is often desirable to use computer simulation to compare alternatives. Several considerations are required for this use. First, it is important to retain independence within each data set (techniques for doing this were discussed in the terminating and steady state sections of this chapter). It is also important that independence is maintained between data sets being compared. This means, in general, that different random number seeds should be used when running the different model scenarios. Other possible correlations can be reduced by using large sample sizes or by ensuring each data point is the average of multiple observations. The Oslen Manufacturing case is continued to illustrate comparing models.

Oslen AGV System (Continued)

After developing a baseline model of existing operations and ensuring output matched the real-world system, Oslen simulation analysts made modifications to the model to represent a new computer system. This effectively scheduled and routed the AGVs according to more sophisticated algorithms that use artificial intelligence to anticipate future load movements based on past patterns. By purchasing new software, the expense of redeveloping the entire system could be avoided. After close consultation with the software vendors, the model was modified and the following data was collected (Table 6.2):

42	56	44	64	45
31	51	60	21	32
38	29	39	64	51
78	32	33	52	53
19	35	37	13	36
62	45	16	57	79
46	61	43	36	57
38	57	34	49	25
42	30	55	27	85
48	89	33	20	70
65	55	34	41	46
55	63	56	40	41
16	56	41	59	34
68	22	11	59	33
51	65	62	60	27
33	68	66	61	33
33	62	73	45	31
35	49	65	68	75
64	50	20	40	40
30	33	40	19	11

Table 6.2 Oslen's New Data Set

The new data set was also found to be normally distributed (see Figure 6.8).

Tests for Normality

Test		--Statistic---		-----p Value------
Shapiro-Wilk	W	0.986916	Pr < W	0.4320
Kolmogorov-Smirnov	D	0.051112	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.043771	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.319491	Pr > A-Sq	>0.2500

Figure 6.8 Normality Statistics for New Data Set

Since data from both runs are normally distributed, a simple t-test can be used to test for differences. Had the data sets not been normally distributed, then a Mann-Whitney test would have been used. The data analysis features of an Excel spreadsheet were used to run the t-test. Figure 6.9 illustrates the outcome.

	Data Set 1	Data Set 2
Mean	42.48	47.35
Variance	234.19	307.04
Observations	100	100
Pooled Variance	270.62	
Hypothesized Mean Difference	0	
Df	198	
t Stat	-2.09	
P(T<=t) two-tail	0.04	
t Critical two-tail	1.97	

Figure 6.9 t-Test Statistics

The mean of the modified simulation is 47.35. This compares to the original mean of 42.48. The t-stat of 2.09 provides a probability less than 4% that the means are not different (p<.04). The simulation analysts at Oslen's Manufacturing concluded the new computer algorithms for their AGV system could result in an approximate 11% gain in productivity. A cost/benefit analysis then could be conducted with these simulation results.

6.4 Output Reporting

Knowing what statistics to gather from a simulation is a very important part of the simulation process. Most models have the capability of providing much more information than is required. The simulation analyst must be able to discern between the summary data that will be useful to the model's customers and the detailed supporting data which verifies that the model is performing as desired. When the statistical work has been completed, results that suit the consumer of the information must be constructed.

6.4.1 Simulation Report

A simulation report is a document intended to communicate the results of a simulation study. A typical simulation report should contain the following elements:

1. Title page – The title page should contain the name of the system being simulated, the date of the report, a revision number or letter, the party for whom the report is being generated, and the name and/or company of the simulation analyst.
2. A short executive summary might be included directly following the title page to provide the important information in a quick, easily understood way.
3. Table of contents – If the report is more than a few pages long, a table of contents should be included.

4. Statement of objectives – The goals and objectives for the simulation study including any specific questions that were to be answered should be summarized in this section of the report.
5. Methodology – This section should include simulation input assumptions and a brief statement describing the general methods used to develop the model. It should be kept in non-technical terminology. The limitations of the model should also be discussed here.
6. Conclusions and recommendations – A summary of the major findings and any recommendations are included in this part of the output report document.
7. Findings – This section of the report should provide data that backs up the "Conclusions and Recommendations". Graphs, tables, and charts that make the data easier to understand and more readable should be used. All data needs to be fully explained.
8. Appendix – The appendix should contain the actual simulation code, all input data, raw output data, reference materials, and calculations.

6.4.2 Quick Reports

Many simulation languages provide special output report generation facilities. These tools can be used to produce graphs and tables, as well as perform statistical analysis. For routine simulation runs that are used by the same customer time after time, a standard format can be developed. The simulation can then produce a report automatically after each production run. An example demonstrating the usefulness of this "quick report generation" can be found in a job shop environment. Each time a new order is received, a simulation is performed to determine how long certain production machines will be in use. The same model is used for each new order with its input parameters set to reflect the quantities and types of parts being built. A standard report is generated by the simulation and copied to the appropriate shop personnel.

6.4.3 Logic Transfer

In some applications, more than an output report is needed to effectively communicate the results of a simulation study. In order to ensure all subtle aspects of a complex system's model are implemented in the real world, a logic transfer should be performed. Logic transfer involves moving information, data, or code from a computer simulation into the real world computer that will be controlling the actual system.

Simulations requiring logic transfer often are being used as design tools. They are precursors to the actual system being built. The simulation team has a goal to identify all logic necessary to implement a controller or other complex logic to be used in system development. This means the simulation team must effectively communicate its developed and debugged algorithms to the system design team. This ensures that the required logic is incorporated into the actual controller for the real system to operate just like the model did. Four methods of facilitating logic transfer have been devised: Philosophic Transfer, Pseudocode Transfer, Database Transfer, and Actual Code Transfer.

Philosophic Transfer: The most common and perhaps least efficient method of transporting simulation logic into actual system control logic is the philosophic transfer. Under this scenario, logic for the real-world system is based on the key ideas and assumptions used in the simulation. This information is presented to the system design team either verbally or in the form of a written report. The design team then evaluates the simulation findings and uses them as a guide or criterion in implementing the actual system software.

In many simple cases, the philosophic transfer works quite well. However, there are some inherent shortcomings that can cause inefficiencies. One of these inefficiencies is duplication of effort. The system software is designed, coded, debugged, and tested. This same process has already been used in the development of the simulation. Another potential shortcoming is the possibility that a piece of information was not communicated properly resulting in either a misinterpreted or omitted portion of logic. A subtle difference in logic can produce a discrepancy that renders the simulation results invalid. The philosophic transfer method tends to make simulation validation (Carson, 1986) a more complex task. Since the two software systems were developed in a somewhat independent fashion, the differences become harder to identify. Whether the simulation accurately depicts the actual system becomes a difficult question with an answer that is hard to prove.

The best way to ensure that these problems don't occur when employing the philosophic transfer method is to form a system design team consisting of both simulators and software engineers. A common design can be arrived at and ideas can be tested with a simulation. When the design is finalized and the simulation is complete, the system software can be written. If the project has emphasized communication and structure, the transfer of the philosophy contained in the simulation should be successful. Duplication of effort has been minimized by doing the initial design work collectively. The ideal philosophic transfer approach to implementing a simulation in the real world is characterized by team work and communication.

Pseudocode Transfer: The second method of transferring data from a simulated controller to an actual controller is the pseudocode transfer method. This approach is very similar to the philosophic transfer but it takes the level of detail a step further. It evolved from the need to ensure that all assumptions incorporated into a simulation were addressed and made apparent to the actual system design team (Figure 6.10). In this scenario, pseudocode is generated by simulation personnel as a method of documenting what has been implemented. This pseudocode is analyzed and rewritten for the actual controller by system software engineers.

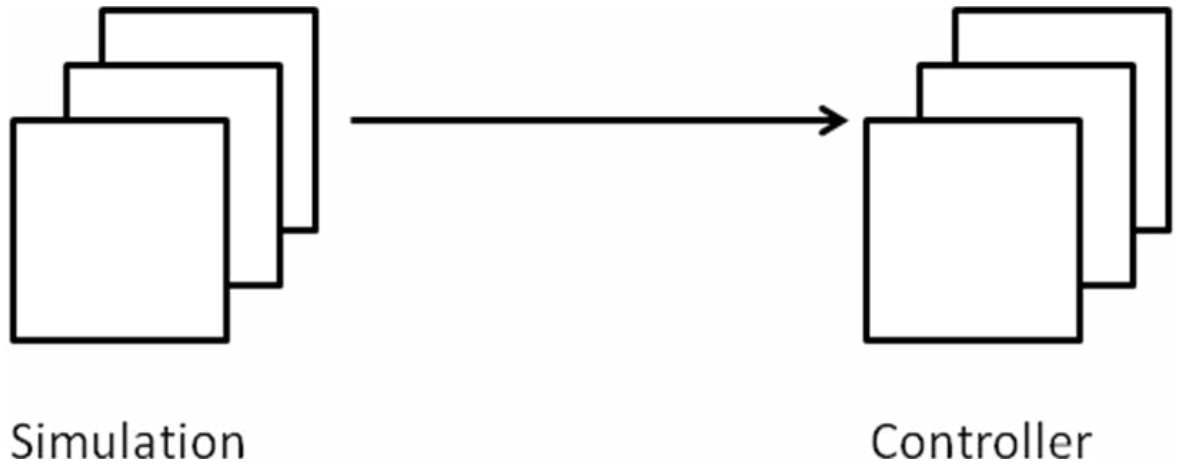


Figure 6.10 Pseudocode Transfer

The pseudocode method of transfer offers the advantage of being more detailed. Therefore, the number of omissions and oversights occurring are expected to be fewer than would be seen when using the philosophic transfer method. Simulation validation is also made easier. The logic rules used in the control system can be examined and verified as identical to those in the simulation. Differences are identified readily and the pseudocode provides a common record for both the simulation and system software package.

A disadvantage associated with using this method of transfer is duplication of programming effort. The pseudocode has to be rewritten in the language of the controller, debugged, and tested. Some of the simulation team's time is also required to rewrite their logic in the form of pseudocode.

Communication between the simulation team and the system design team is important under this transfer scenario; but not nearly as crucial as when using the philosophic transfer method. By placing a structured and detailed pseudocode document in the hands of the design team, little additional interaction will be required of the simulators. It is not until the verification and validation phase of the project, when the completed system software is compared to the simulation, that further communication will be required.

Database Transfer: The third method of transferring simulation logic to an actual system is known as a data base transfer. The essence of this transfer method is to transport a data base, developed in the simulation, and use it to drive the actual system controller. It is important to note that this method of data transfer is contingent upon software existing in both the simulation and controller. This requires that an initial development project must be undertaken to define the database and delineate how it will be used. After this has been established and the simulation and controller software have been developed, the data base can be transferred and utilized. This type of transfer will most commonly be used in cases where many similar systems are being designed using a generic simulation and a generic controller.

The advantages to using this transfer method are quite apparent. The debugging of the data base and its testing are all done in the simulation phase. Nearly all duplication of programming effort is eliminated (after the initial system has been created). Changes to the simulation and actual system can be done easily and consistently. Simulation validation and verification becomes much easier. In addition, communication and the passing of abstract concepts becomes less of an issue.

A potential drawback encountered when using the data base transfer method is the loss of flexibility. Although using a data base simplifies logic and provides the system engineers with a standard, unusual cases and exceptions become more difficult to incorporate. For this reason it is very important that the data bases be designed in a comprehensive manner.

Actual Code Transfer: The final method of data transfer to be examined is transporting actual code from simulation to system controller. This can be done either when the simulation has been written in a conventional language such as C# or when using a specialized simulation language such as GPSS/H or GPSS/PC which has the capability of calling external subroutines. These subroutines contain code that will be used in the system controller. The FORTRAN subroutines can be written either by the simulators or by the system design team prior to implementation in the simulation. Duplicated effort is reduced because the control logic is written only once in the simulation phase and then reused in the actual system.

Using this actual code transfer method requires that the simulators and system designers work very closely. It may be advantageous to form one team consisting of both simulation and design personnel. A disadvantage inherent when using this method is the requirement of a more concerted initial effort and additional time to debug the model. The simulator may not be familiar with the control algorithm and the system designer may not be familiar with the rest of the simulation. When the model is run and debugged, some learning time might be required. However, testing and debugging the simulation is also testing and debugging the actual system.

6.5 Post Processing Output

All models must provide some type of output. This output can be communicated or presented to the end user in many different ways. Traditionally, simulation output data took the form of hard to understand stacks of numerical data. Some simulation analysts thrive on reams of paper covered with statistics. Coupled with this overkill tendency of the analysts, is the ease with which most simulation languages allow data to be gathered and reported. Nearly all simulation software products automatically make an abundance of data available at the end of each run.

Knowing what data to report and what data to omit are challenges for the simulation analyst. A good rule of thumb is to make the data as readable as possible and limit the quantity to the essentials. Two types of output data are usually available from a simulation. The first type describes the model itself and will help the analyst verify the model. This does not need to appear in a simulation report. The second type of data is the output that describes characteristics of the system being modeled. This data will be of interest to the end user and should appear in some type of output report. Tables, graphs, and bar charts can all be used to make the raw data more readable.

Although many simulation products contain data analysis tools, it may still be desirable to explore some commercially available software. The reason for this is two-fold. First, the function of a simulation software product is usually modeling, not data presentation. Second, excellent data presentation software is available. This software has been developed specifically for putting data into a form that can be easily understood and communicated. The simulation analyst can use these types of packages to great advantage. Many different data analysis and presentation software packages are currently available. Two of these are briefly profiled in sections. 6.5.1 and 6.5.2.

6.5.1 MS-Office Excel

A very popular method of transforming raw simulation data into an easy-to-read report is through the use of spreadsheet software. Many spreadsheet programs are currently available from a wide variety of sources. The most popular of these is MS-Office Excel. Spreadsheets are programs which allow the user to manipulate data that has been entered into a series of cells organized into rows and columns. Although it is possible to enter the data into the spreadsheet from the computer keyboard, the simulation analyst will probably import a raw simulation output data file into the spreadsheet. Excel provides the capability of performing this function in a variety of ways with various data import tools. Many of these can be located on the data ribbon (see Figure 6.11).

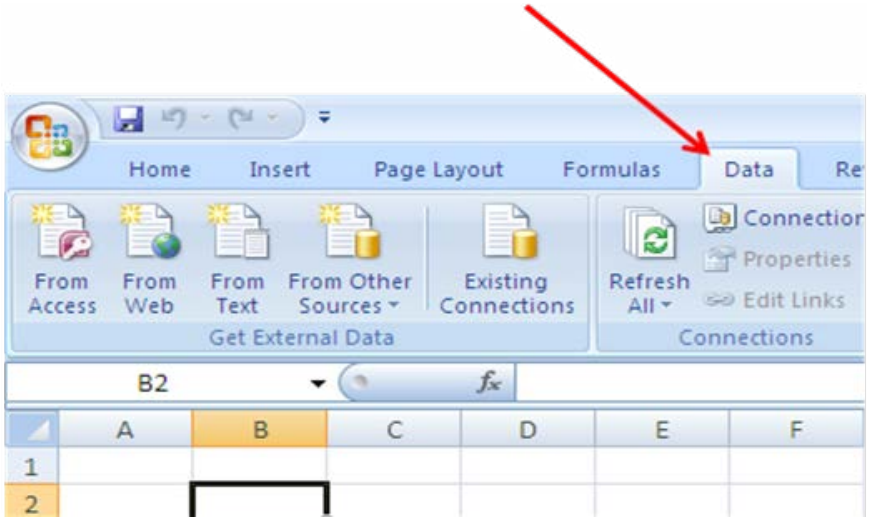


Figure 6.11 Data Ribbon in MS-Excel

The first three icons on the data ribbon provide capability to import data from MS-Access, Web pages, and text files. Many simulation products create output files that can be imported using the text option. This involves browsing out to the folder where the file is located, then selecting it for import.

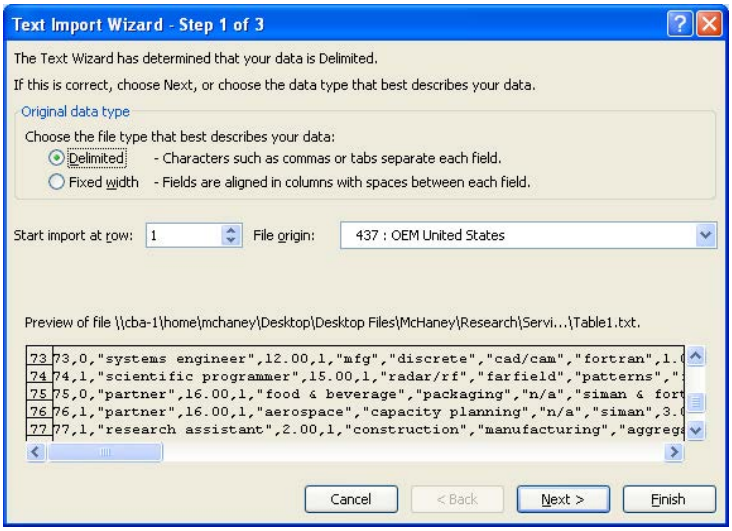


Figure 6.12 Importing Data in MS-Excel

Excel uses a wizard that will enable the analyst to select the delimiting character (space, comma, et cetera) and then import the data for analysis (see Figure 6.12). Excel also allows the import of data tables from Web pages. Using the 'From Web' button launches a browser window. The analyst can paste the URL where the data resides and a list of eligible tables will be displayed (See Figure 6.13). Clicking on a table will import it into Excel (see Figure 6.14).

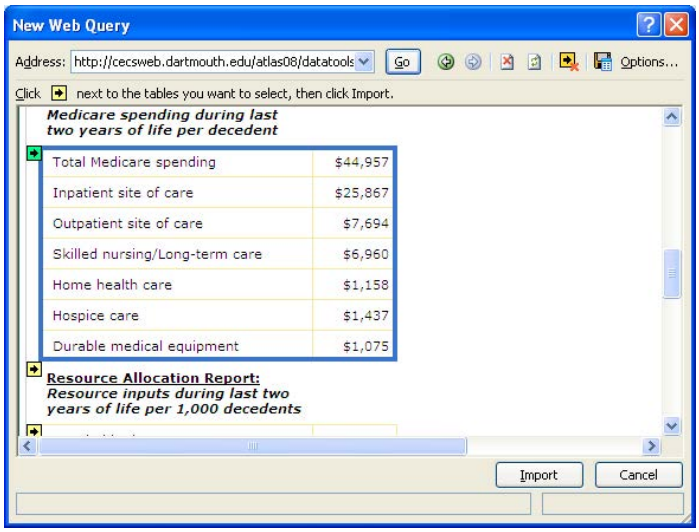


Figure 6.13 Web Query in MS-Excel

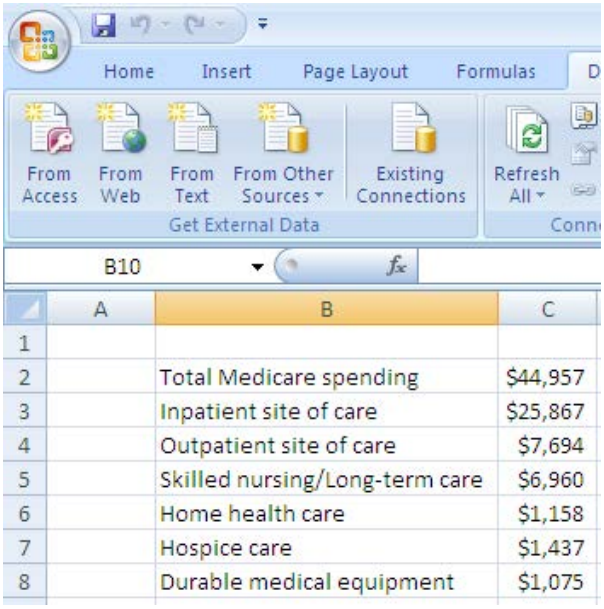


Figure 6.14 Data from a Web Query in MS-Excel

Once the data has been transported into the spreadsheet, any of the data manipulations offered by the software may be performed. For instance, Excel provides a wide variety of statistical analysis tools. Figure 6.15 looks at the data analysis menu showing some of the selections.

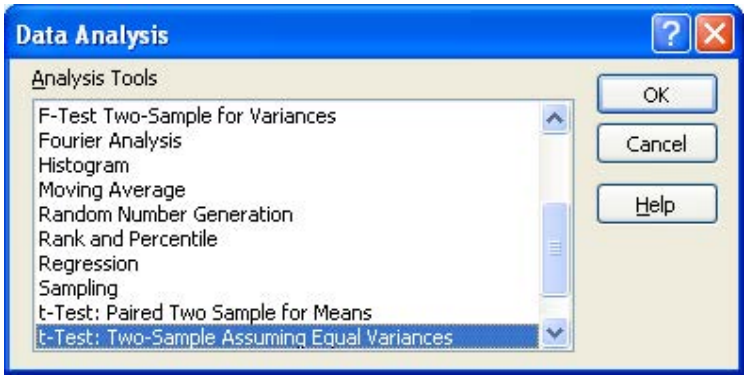


Figure 6.15 Data Analysis Menu in MS-Excel

6.5.2 SAS

Another commercially available software package that can be used as a simulation output data post processor is SAS 9.2 Software from SAS. SAS 9.2 is a powerful data analysis environment that permits statistical experiments to be conducted on sets of data. Fully capable of performing traditional statistical tests and the nonparametric tests often required in simulation analysis, SAS can take most simulation output beyond what is possible with the simulation software packages. Figure 6.16 provides a look at SAS running an analysis.

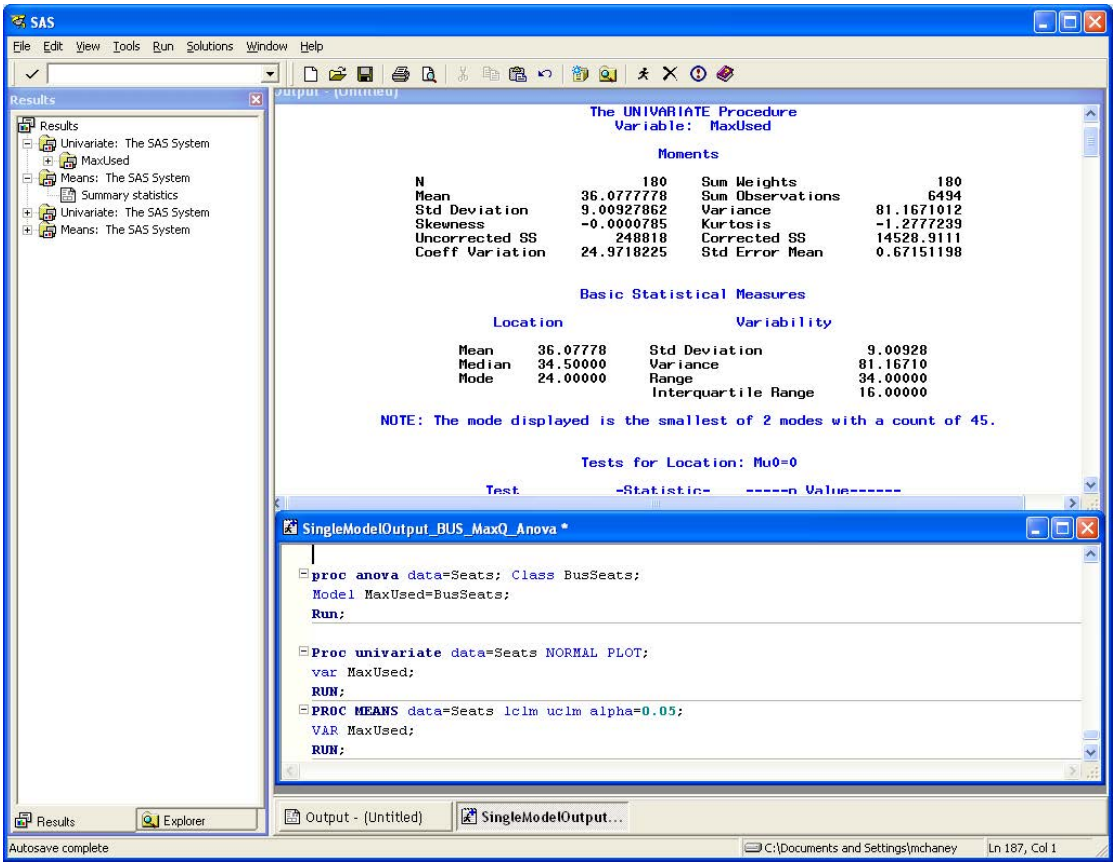


Figure 6.16 SAS Program and Output

6.5.3 Custom Generated Animation

Using animation software with computer simulation is discussed in Chapter Two. Most simulation products include animation software or third party software (such as Proof), are available to take full advantage of current graphics technology and produce exciting high resolution animation to accompany simulation program output.

6.6 Operations, Maintenance and Archival Phase

This phase involves storing the model with consideration of its potential future use. Some models are not stored but rather are used in decision support. For instance, a model may be used for staffing questions and for daily use to determine the expected impact of changing conditions. In other cases, the model's current need is fulfilled and it should be stored, maintained, and the project team disbanded.

Another common activity at this point in time is conversion. This may not apply to all situations but it may involve turning the model over to the customer or end-user. These end-users may adopt the simulation as a tool requiring little or no additional intervention from the simulation team. The storage component of the archive phase reflects the investment in the simulation project. Documentation and physical simulation code, as well as data and other records from the project should be stored in a secure, safe place for future access. Simulations are becoming important repositories for corporate knowledge and can provide important inputs to future organizational decision making.

The third component in the archive phase is maintenance. Here the simulation is monitored and adjusted to reflect environmental changes or newly available organizational information. Often, a member of the simulation project team will be appointed as the contact person for maintenance issues.

The final component in the archive phase is the dissolution of the project team. When the full team is no longer warranted, it is dissolved and the team's members freed for other duties.

6.7 Bibliography

J.M. Bloodgood and R.W. McHaney, "DSS-Supported Knowledge Acquisition and Transfer: An Exploration," Journal of Information and Knowledge Management. Vol. 2, No. 3, 1–10 (2003).

C.W. Emory, Business Research Methods, Third Edition. Richard D. Irwin, Inc., Homewood, Illinois (1985).

J.O. Henriksen and R.C. Crain, GPSS/H User's Manual, Wolverine Software Corporation, Annandale, Virginia (1983).

F.N. Kerlinger, Foundations of Behavioral Research, Harcourt, Brace, Jovanovich College Publishers, Fort Worth, Texas (1986).

J. Kleijnen, Statistical Tools for Simulation Practitioners. Marcel Drekker, New York (1987).

A.M. Law and W.D. Kelton, Simulation Modeling and Analysis, 3rd Edition, McGraw-Hill Book Company (2000).

R.W. McHaney, Computer Simulation: A Practical Perspective, Academic Press, San Diego (1991).

R.W. McHaney, "Reusing Simulation Logic in System Development Projects," Progress in Simulation, Volume I (George Zobrist and James V. Leonard, editors). ABLEX Publishing Corp. Norwood, New Jersey, 159–185 (1991).

R.W. McHaney, Simulation, In Miriam Drake (Ed.), Encyclopedia of Library and Information Science, Second Edition, Marcel Dekker, New York, 2643–2655 (2003).

R.W. McHaney, "Success Surrogates in Representational Decision Support Systems," Encyclopedia of Information Science and Technology, Vol 1 (Mehdi Khosrow-Pour, ed.), Idea Group Publishing, Hershey, PA, 2672–2677 (2005).

R.W. McHaney and D. White, Discrete Event Simulation Software Selection: An Empirical Framework" Simulation & Gaming, 29(2) 228–250 (1998).

R.W. McHaney, D. White and G. Heilman, "Simulation Project Success and Failure: Survey Findings," Simulation & Gaming, 33(1), 49–66 (2002).

J. McLeod, Computer Modeling and Simulation: Principles of Good Practice 10(2). Simulation Councils, Inc., La Jolla, California (1982).

M. Page-Jones, The Practical Guide to Structured Systems Design, Yourdon Press, New York (1980).

E.J. Pedhazur and L.P. Schmelkin, Measurement, Design, and Analysis: An Integrated Approach, Erlbaum, Hillsdale, NJ (1991).

S. Robinson, R.E. Nance, R.J. Paul, M. Pidd and S.J.E. Taylor, "Simulation Model Reuse: Definitions, Benefits and Obstacles," Simulation Modelling Practice and Theory, Elsevier, 12, 479–494 (2004).

D. White and R.W. McHaney, Simulation Steps, Working Paper (2009).

7 Case Study: DePorres Tours

Chapter 7 focuses on taking a simulation project from start to finish. Although the system might seem simplistic and could probably be adequately modeled using a spreadsheet, the intent is to demonstrate the thought process used in defining, analyzing, and applying modeling principles. The system to be analyzed provides a good framework for discussion and will be interesting to a wide variety of simulation (or potential simulation) analysts. The project will follow the simulation life cycle illustrated in Chapter 4 (See Figure 7.1).

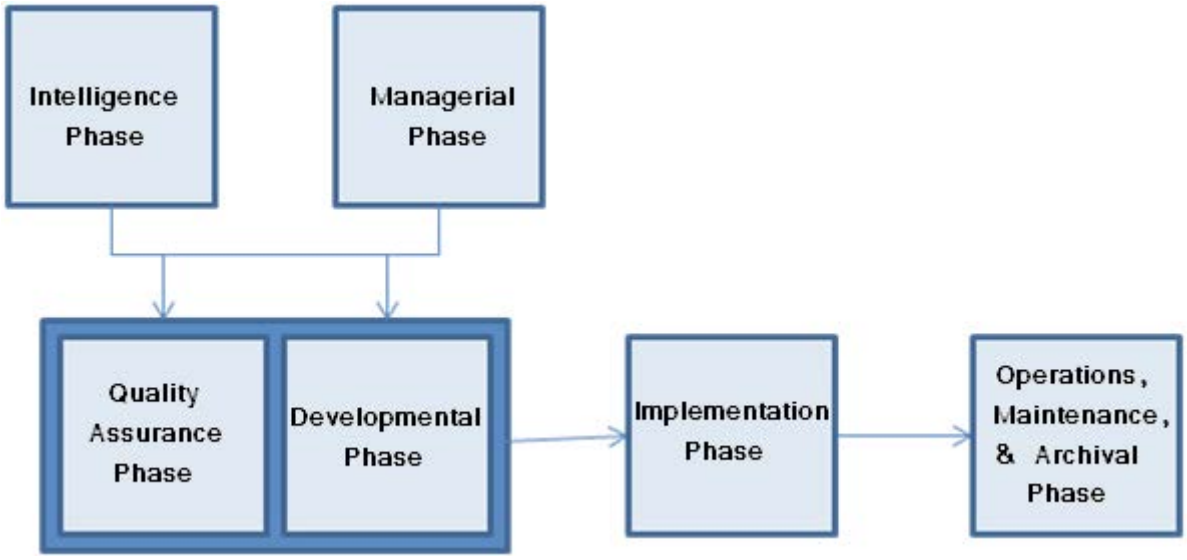


Figure 7.1 Phases in the Simulation Life Cycle

7.1 Intelligence Phase

The intelligence phase of the simulation life cycle involves understanding the environment and determining problems to be solved. This often starts with a general understanding of the system to be modeled together with a problem definition and feasibility considerations.

DePorres Tours: General Scenario

DePorres Tours provides a small minibus shuttle system in downtown Chicago. In general, they sell customers a day pass that allows access to a minibus as it shuttles between five major stops in the city. The pass is unlimited for the day, enabling tourists to disembark, enjoy visiting a Chicago landmark at their leisure, then re-enter the bus and shuttle to another location. The minibus stops at 5 locations: Stop 1: The new Millennium Park, Stop 2: Lincoln Park Zoo and Gardens, Stop 3: The Sears and Hancock Towers, Stop 4: The Magnificent Mile, Stop 5: Navy Pier.

7.1.1 Problem Definition

DePorres Tours wants to create a simulation to help determine bus capacity for their tour operation and then be able to utilize the model in the future to pre-test any changes to their operation. Their current problem statement becomes:

"What size bus will best accommodate expected customer traffic?"

Specifically, they are looking at buying either a 24 or a 48 passenger bus.

7.1.2 General Feasibility

Before beginning the simulation project, DePorres assessed general feasibility using the TELOS approach (See Figure 7.2):

Technical – A staff member took a simulation class and has the technical skills needed to develop a model. Current available simulation software is certainly capable of modeling a system such as the tour bus route.

Economic – A preliminary cost study indicated software and wages will not be excessive and will meet budget constraints.

Legal – No legal issues will result from developing and using this model.

Operational – The model will be routine and the developed system does not appear to have any operational issues. An accurate model can be developed.

Schedule – Model development is not under a strict timeline and should meet the required purchase date for the new bus.

Figure 7.2 Feasibility Questions

7.2 Managerial Phase

DePorres Tours recently sent Telly O'Sullivan to a simulation training course and has allocated 4 hours a day for him to gather information and conduct the simulation project. Since it is his first project, he has been given a month to complete the model. DePorres Tours owner, the indomitable Kafy DePorres has informed all drivers, tour guides, and office personnel about the project and strongly encouraged their full cooperation. Additionally, a small team of one driver, one tour guide, and Kafy herself will meet with Telly weekly to assess his progress and help with any problems. Kafy has set aside adequate funds for the project and plans to acquire a simulation software package for model development.

7.3 Developmental Phase

Telly, anxious to apply what he learned in class, developed a working view of the system to be modeled. He used the following definition to help break the tour bus system into its major components.

System: A set of components or elements that are related to each other in such a manner as to create a connected whole.

He saw the system as having these related parts (Table 7.1):

Bus
Bus Driver
Tourist
Bus Route
Bus Stops

Table 7.1 Related Subsystems

He also defined relationships between elements (See Table 7.2)

Entity	Bus	Bus Driver	Tourist	Bus Route	Bus Stops
Bus		Bus operated by Driver	Bus ridden by Tourist	Bus follows Bus Route	Bus stops at Bus Stops
Bus Driver	Bus Driver operates Bus		Bus Driver checks Tourist ticket	Bus Driver guides Bus along Route	Bus Driver interacts with Tourists at Stops
Tourist	Tourist rides Bus	Tourist shows ticket to Driver		Tourist follows Bus Route while on Bus	Tourists embarks and disembarks at Bus Stops and interacts with Driver at Stops
Bus Route	Bus Route followed by Bus	Bus Route followed by Driver	Bus Route followed by Tourist		Bus Route incorporates Stops
Bus Stops	Bus Stops are used by Buses	Bus Driver halts Bus at Stops	Bus Stops are where tourists get on and off bus	Bus Stops are along the Bus Route	

Table 7.2 Relationships between Subsystem Elements

Although the table contains some repetitive information, it becomes a working document to help understand the system and its basic elements.

7.3.1 Environment and Boundary

Based on the data shown in Table 7.2, the system environment can be defined as the road system in Chicago, existing traffic patterns, and tourism patterns. All these things lie outside the system and have an influence on its behavior. The systems boundary separates the tour bus system, together with tourists, driver, route, and stops from the environment.

7.3.2 Model Scaling and Scope

Telly decided the scope and scale of his model would include the following:

Scope:

1) Customer arrivals and departures will be modeled at each bus stop.
2) Bus driver and bus will be modeled.
3) Bus travel times along the route will be modeled.

Scale:

1) Two customer types will be considered: first time riders and those using the bus as a taxi.
2) Bus driver and bus will be modeled as a single resource. Bus driver behavior will be averaged into bus stop times. In the future this could be redeveloped using bus driver persims.
3) Bus travel times will be modeled using collected distributions covering an entire day.

7.3.3 Modeling Views

Telly decided to use a process orientation view of the system. In other words, he will view the bus system as a time ordered sequence of interrelated events separated by passages of time. This will influence his choice of simulation software but is consistent with the recent training he received.

7.3.4 Concept Model

Telly created a couple of concept models to help understand the system better. First he created a spreadsheet that calculated hourly capacity and average route time (Figure 7.2).

Google docs roger.mchane

DePorres Tours

File Edit View Format Insert Tools Form Help

	Stop #	AverageTravel Time	Average Time per Stop (Min)	Total Trip Time (Min)			Minutes
3	1	11	3	14.00		Average Time Passenger on Bus	20
4	2	7	3	10.00		Seats on Bus	24
5	3	13	3	16.00			
6	4	16	3	19.00			
7	5	11	3	14.00			
8							
9		12	3	73.00			
10							
11	Total Capacity			87.6			

Figure 7.2 Concept Model with Google Docs

Next he used an Excel spreadsheet with Paul Jensen's ORMM Queuing Add-ins to create a rough model of the bus route if viewed as a queuing system (See Figure 7.3).

Queue Station	Bus_Seats
Arrival Rate	84
Service Rate/Channel	4
Number of Servers	24
Max. Number in System	***
Number in Population	***
Type	M/M/24
Mean Number at Station	23.97426
Mean Time at Station	0.285408
Mean Number in Queue	2.974259
Mean Time in Queue	0.035408
Mean Number in Service	21
Mean Time in Service	0.25
Throughput Rate	84
Efficiency	0.875

Figure 7.3 ORMM Queuing Add-in for MS-Excel

Both models created quick approximations of the system. However, neither sufficiently captured the dynamics of the system adequately. This left Telly with no option other than to move ahead with a full simulation. Both concept models demonstrated a 24 passenger bus could move around 80-84 people per hour but this was in contrast to what had been observed in the real world. Telly suspected the variance in the system had not been included in spreadsheets realistically.

7.3.5 Model Inputs

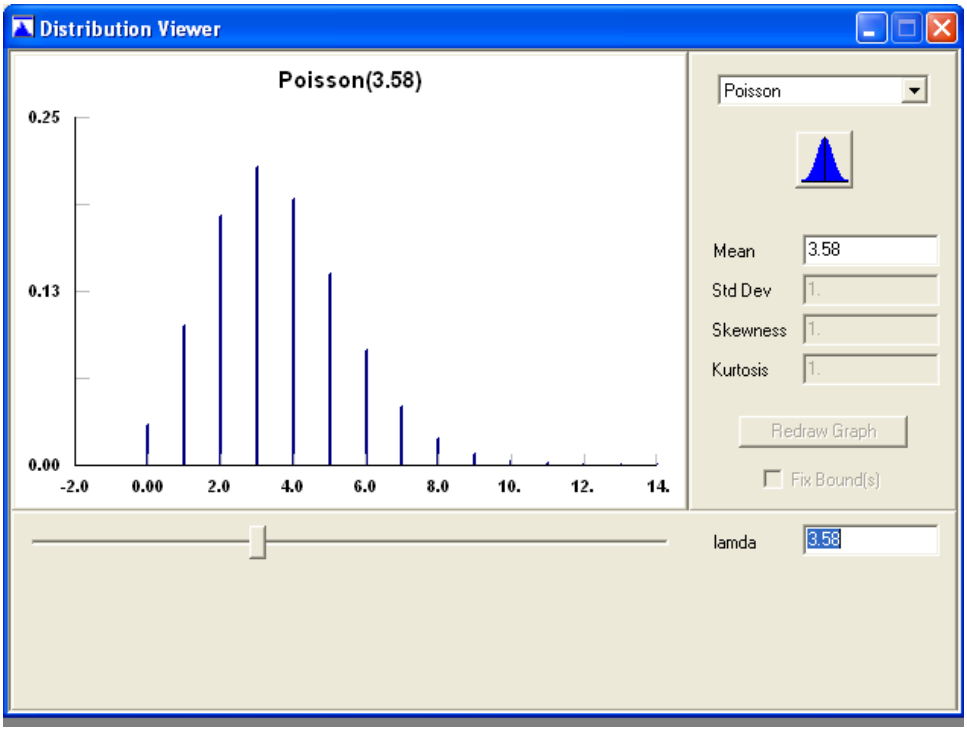
Telly's next task was to begin data collection in earnest. Although he had calculated quick averages he used in the concept models, he wanted to develop more accurate information. He did this by visiting each bus stop and collecting customer arrival times as they waited for the bus. Table 7.3 provides a look at the data he collected at Bus Stop #1. He loaded this into Stat::Fit 2 and determined the tourists were arriving according to a Poisson distribution with a mean of 3.58 minutes (See Figure 7.4).

1	3	5	4	5
5	1	3	4	4
5	2	2	5	1
3	4	4	6	5
3	3	1	4	2
3	3	6	4	4
1	9	3	7	1
4	3	6	1	1
5	9	5	3	1
2	6	2	0	5

Table 7.3 Tourist Interarrival Times in Minutes

Figure 7.4 Customer Interarrival Data Fit to a Poisson Distribution

Telly repeated his data collection for the other 4 stops and determined the following interarrival rates (see Table 7.4).



Stop #	Distribution	Mean	Spread
1	Poisson	3.58	n/a
2	Poisson	3.22	n/a
3	Uniform	4	2
4	Poisson	3.34	n/a
5	Poisson	3.12	n/a

Table 7.4 Tourist Interarrival Times

Telly continued to collect input data throughout the week and was able to determine the following quantitative information would need to be included in the model:

Average Time Bus Spends at a Stop: Normally distributed 2 minutes with a standard deviation of .3 minutes.

Time between Stops (see Table 7.5).

Time Between Stop #	Distribution	Mean / Spread (seconds)
1–2	Exponential	12,4
2–3	Exponential	8,3
3–4	Exponential	14,4
4–5	Exponential	8,3
5–1	Exponential	15,4

Table 7.5 Time per Bus Routes

- Passenger Route Decision Percentages (see Table 7.7).

Stop # Destination %	1	2	3	4	5
1	0	80	10	5	5
2	2	0	76	20	2
3	4	2	0	67	27
4	15	6	3	0	76
5	88	8	2	2	0

Table 7.7 Destination Percentages

- Qualitative Data. The following assumptions were also gathered:

1) The bus always goes in sequence 1-2-3-4-5 then back to 1.
2) The bus stops even if no passengers are visible at the stop or desire to be let off.
3) Bus drivers change shifts without disrupting the schedule.

7.3.6 Simulation Input Data Validation

As part of the Quality Phase of his simulation project, Telly validated his input data in the following ways:

Observation: He observed the current system and double checked his input data against what he observed.
Expert's Opinions: He showed the current drivers and ticket sellers his assumptions and input data for evaluation.

Intuition and Experience: He also used his own experience to double check the data.
Telly continued to check for validation throughout his model development process.

7.3.7 Selection of a Language or Tool

Telly was now ready to select a simulation tool for model development. He initially looked into both simulation languages and simulator packages. After doing a little research and since he had recently taken a GPSS simulation class, he decided to concentrate on general purpose simulation languages. Telly used a standard approach to the software evaluation process. The four steps in his procedure were:

1) Define needs of project
2) Research market
3) Compare available options
4) Select software

What will the frequency of future simulation work be? Will this be a onetime project, or will simulation be used regularly from now on? The bus simulation will be a onetime project. It will be maintained so that it might be modified and used for future analysis.

Step one: Define needs – Telly began the software selection process by analyzing the needs of his current simulation project and the anticipated needs of his future simulation projects. He did this by answering some questions.

What type of system will be modeled? Is it unique in function? A unique system is being modeled but a simulator package or language could be located to perform the task.

Who will be doing the simulation work? Telly would be doing the work himself. He was comfortable with his new computer skills and desired to use a full power simulation language.

What budgetary constraints exist? Telly had already obtained permission to purchase a software package within reason and had access to a computer system. So, budgetary constraints were not really a major consideration for this project.

Who will be using the results? The results of the study would be used by Kafy DePorres as a means of analyzing the bus route and making a purchase.

Step two: Research – Telly compiled a list of several simulation software packages including the one he had recently trained on – GPSS/World.

Step three: Compare – Telly developed an evaluation template to help compare available options in the software selection process. The weighting factors were subjectively developed from attributes shown in Figure 7.5.

	Weighting Factor		Product Score		Total
Graphics	1	X	____	=	____
User-Friendliness	7	X	____	=	____
Software capability	7	X	____	=	____
Ease of Use	7	X	____	=	____
Power	5	X	____	=	____
Cost	6	X	____	=	____
Output reports	9	X	____	=	____
Ease of Debugging	8	X	____	=	____
Statistics	7	X	____	=	____
Customer Support	6	X	____	=	____
Training	10	X	____	=	____
Documentation	7	X	____	=	____
Vendor stability	4	X	____	=	____
Application specific modules	1	X	____	=	____
			Total Product Score =		____

Figure 7.5 Simulation Software Evaluation Template

Step four: Make Selection – Telly used his template to evaluate four simulation products and decided, largely due to his training, that GPSS/World would be most suited to his current application.

7.3.8 Model Construction

Telly created his model using GPSS World. He broke his model into several segments to represent different elements of the system and initially used a block diagram to help conceptualize the code (Figure 7.6).

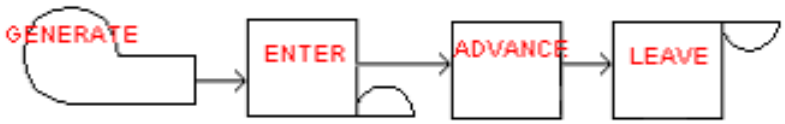


Figure 7.6 GPSS Block Diagram

The first segment represented the Bus Route (see Figure 7.7).

```
**********************************************************************
*  Bus Route With Starts and Stops and Waits for Passengers
**********************************************************************
         GENERATE   ,,,1                    ;Create Bus Transaction

Again    ADVANCE    (Exponential(1,12,4))   ; Bus Travels to Stop #1
         ENTER      STOP1                   ; Bus at Stop
         ADVANCE    (Normal(2,2,.3))        ; Passenger Loading/Unloading Time
         LEAVE      STOP1                   ; Leave Bus Stop

         ADVANCE    (Exponential(3,8,3))    ; Bus Travels to Stop #2
         ENTER      STOP2                   ; Bus at Stop
         ADVANCE    (Normal(4,2,.3))        ; Passenger Loading/Unloading Time
         LEAVE      STOP2                   ; Leave Bus Stop

         ADVANCE    (Exponential(5,14,4))   ; Bus Travels to Stop #3
         ENTER      STOP3                   ; Bus at Stop
         ADVANCE    (Normal(6,2,.3))        ; Passenger Loading/Unloading Time
         LEAVE      STOP3                   ; Leave Bus Stop

         ADVANCE    (Exponential(7,8,3))    ; Bus Travels to Stop #4
         ENTER      STOP4                   ; Bus at Stop
         ADVANCE    (Normal(8,2,.3))        ; Passenger Loading/Unloading Time
         LEAVE      STOP4                   ; Leave Bus Stop

         ADVANCE    (Exponential(9,15,4))   ; Bus Travels to Stop #5
         ENTER      STOP5                   ; Bus at Stop
         ADVANCE    (Normal(10,2,.3))       ; Passenger Loading/Unloading Time
         LEAVE      STOP5                   ; Leave Bus Stop
         TRANSFER   ,Again                  ; Move to Travel Back to Stop #1
```

Figure 7.7 GPSS Code for Bus Route

The second segment represented the passengers and their decision regarding where to get off the bus.

```
*************************************************************************
         GENERATE   (POISSON(11,3.58))          ; Passengers at Stop 1
         QUEUE      STOP1Q                      ; Wait for Bus
         TEST E     BV$STOP1V,1                 ; Bus Arrives, If Space Get Onboard
         ENTER      BusSeats                    ; Sit in Seat
         DEPART     STOP1Q                      ; No Longer in Queue
         Transfer   .8,Next1a,OutAt2            ; Some go to Stop 2
Next1a   Transfer   .5,Next1b,OutAt3            ; Some go to Stop 3
Next1b   Transfer   .5,OutAt4,OutAt5            ; Some go to Stop 4 or 5

         GENERATE   (POISSON(12,3.22))          ;Passengers at Stop 2
         QUEUE      STOP2Q                      ; Wait for Bus
         TEST E     BV$STOP2V,1                 ; Bus Arrives, If Space Get Onboard
         ENTER      BusSeats                    ; Sit in Seat
         DEPART     STOP2Q                      ; No Longer in Queue
         Transfer   .76,Next2a,OutAt3           ; Some go to Stop 3
Next2a   Transfer   .834,Next2b,OutAt4          ; Some go to Stop 4
Next2b   Transfer   .5,OutAt5,OutAt1            ; Some go to Stop 5 or 1

         GENERATE   12,2                        ;Passengers at Stop 3
         QUEUE      STOP3Q                      ; Wait for Bus
         TEST E     BV$STOP3V,1                 ; Bus Arrives, If Space Get Onboard
         ENTER      BusSeats                    ; Sit in Seat
         DEPART     STOP3Q                      ; No Longer in Queue
         Transfer   .67,Next3a,OutAt4           ; Some go to Stop 4
Next3a   Transfer   .818,Next3b,OutAt5          ; Some go to Stop 5
Next3b   Transfer   .67,OutAt1,OutAt2           ; Some go to Stop 1 or 2

         GENERATE   (POISSON(13,3.34))          ;Passengers at Stop 4
         QUEUE      STOP4Q                      ; Wait for Bus
         TEST E     BV$STOP4V,1                 ; Bus Arrives, If Space Get Onboard
         ENTER      BusSeats                    ; Sit in Seat
         DEPART     STOP4Q                      ; No Longer in Queue
         Transfer   .76,Next4a,OutAt5           ; Some go to Stop 5
Next4a   Transfer   .625,Next4b,OutAt1          ; Some go to Stop 1
Next4b   Transfer   .67,OutAt2,OutAt3           ; Some go to Stop 2 or 3

         GENERATE   (POISSON(14,3.12))          ;Passengers at Stop 5
         QUEUE      STOP5Q                      ; Wait for Bus
         TEST E     BV$STOP5V,1                 ; Bus Arrives, If Space Get Onboard
         ENTER      BusSeats                    ; Sit in Seat
         DEPART     STOP5Q                      ; No Longer in Queue
         Transfer   .88,Next5a,OutAt1           ; Some go to Stop 1
Next5a   Transfer   .67,Next5b,OutAt2           ; Some go to Stop 2
Next5b   Transfer   .5,OutAt3,OutAt4            ; Some go to Stop 3 or 4
```

Figure 7.8 GPSS Code for Passengers

The third segment represents the passengers leaving the bus when it arrives at their expected stop.

```
OutAt1   QUEUE      WAIT41                ; Tracking Travel Time
         GATE SF    STOP1                 ; Passenger Waits for Stop
         LEAVE      Busseats              ; Gets out of Seat
         DEPART     WAIT41                ; Leaves Bus
         TERMINATE                        ; Is No Longer in Model

OutAt2   QUEUE      WAIT42                ; Tracking Travel Time
         GATE SF    STOP2                 ; Passenger Waits for Stop
         LEAVE      Busseats              ; Gets out of Seat
         DEPART     WAIT42                ; Leaves Bus
         TERMINATE                        ; Is No Longer in Model

OutAt3   QUEUE      WAIT43                ; Tracking Travel Time
         GATE SF    STOP3                 ; Passenger Waits for Stop
         LEAVE      Busseats              ; Gets out of Seat
         DEPART     WAIT43                ; Leaves Bus
         TERMINATE                        ; Is No Longer in Model

OutAt4   QUEUE      WAIT44                ; Tracking Travel Time
         GATE SF    STOP4                 ; Passenger Waits for Stop
         LEAVE      Busseats              ; Gets out of Seat
         DEPART     WAIT44                ; Leaves Bus
         TERMINATE                        ; Is No Longer in Model

OutAt5   QUEUE      WAIT45                ; Tracking Travel Time
         GATE SF    STOP5                 ; Passenger Waits for Stop
         LEAVE      Busseats              ; Gets out of seat
         DEPART     WAIT45                ; Leaves Bus
         TERMINATE                        ; Is No Longer in Model
```

Figure 7.9 GPSS Code for Departing Passengers

The fourth segment tracks model timing.

```
**********************************************************************
    GENERATE   600            ; 10 Hour Shift in Minute
    TERMINATE  1              ; Stop for Statistics after Each Shift
**********************************************************************
```

Figure 7.10 GPSS Timing Transaction

7.4 Quality Phase

In order to ensure the model was representative of its real world equivalent, Telly spent time on verification. First, during model construction and earlier activities he used preventative verification to ensure the model would not have many errors. He documented his code and used a structured approach to carefully organize the model. He tested individual sections of the code to ensure they operated as expected.

Telly also used appraisal verification to check the programming after it was been coded. He stepped through the model using GPSS World's step function to make sure the model operated as expected. He ran the model under simplified conditions to make sure it operated as expected. Finally, he checked the output reports and looked for any anomalies that might indicate a problem. After a few minor fixes, he was satisfied the model operated correctly.

7.5 Implementation

After having conceptualized, coded, verified, and validated his model, Telly was ready to start running a set of experiments to answer the primary questions about the system – in this instance, what size bus should be purchased. Telly initialized key parameters in the model and set up production runs so he could make inferences about the behavior of the system. He used the sequence of steps shown in Figure 7.11:

Figure 7.11 Experimentation Process

7.5.1 Experimental Design

The primary value of interest for Telly's simulation is "Size of Bus." This value can be obtained in several different ways. Telly decided to run his model with a variety of bus seat capacities and then examine related queuing times to determine the impact of using a smaller bus. Additionally, he set the following experimental conditions:

- Length of simulated run time – Telly used 10 hours for his model run time. The model matches th real world run time and is set to terminate after 10 hours of simulated time.
- Replications – Telly wanted to gather enough data to develop a 95% confidence interval. He decided to start with 30 replications which represents a month of 10 hour days.
- Reseeding random number streams – To ensure independence, Telly reset and cleared the model at the end of each 10 hour day. Each new day used fresh random number streams to ensure the same numbers were not reused for subsequent runs.
- Initial conditions – No special initial conditions were placed into the model. The people started to arrive and the bus began its circuit.

7.5.2 Statistical Output Analysis of a Single Model

Telly knew that the stochastic nature of his model required enough replications to ensure output variability was accounted for in terms of a mean and standard deviation. He wanted to construct a 95% confidence interval within which the true values of his model are likely to fall. He collected the following "Maximum Seat Capacity Required" based on a 64 seat bus. The output data failed the normality test so Telly was forced to use a larger sample size to more confidently capture all variance (Figure 7.12).

Tests for Normality

Test		--Statistic---	-----p Value------	
Shapiro-Wilk	W	0.834894	Pr < W	0.0003
Kolmogorov-Smirnov	D	0.230424	Pr > D	<0.0100
Cramer-von Mises	W-Sq	0.24772	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	1.528084	Pr > A-Sq	<0.0050

Figure 7.12 Failed Test for Normality on Small Data Set

Telly changed to 45 replications and retested for normality. The new Shapiro-Wilk W of .965 does not reject the null hypothesis that the variable is normally distributed (p<.1889). Likewise, Kolmogorov-Smirnov, Cramer-von Mises, and Anderson-Darling tests do not reject the null hypothesis. Additionally, the plots (shown in Figure 7.13) indicate normality cannot be rejected.

Tests for Normality

Test		--Statistic---	-----p Value------	
Shapiro-Wilk	W	0.964989	Pr < W	0.1889
Kolmogorov-Smirnov	D	0.11801	Pr > D	0.1154
Cramer-von Mises	W-Sq	0.058065	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.375992	Pr > A-Sq	>0.2500

Figure 7.13 Test to Establish Normality with Larger Sample Size

He further checked with a Stem and Leaf Plot, a Boxplot, and a Normal Probability Plot. All indicated normality (See Figure 7.14).

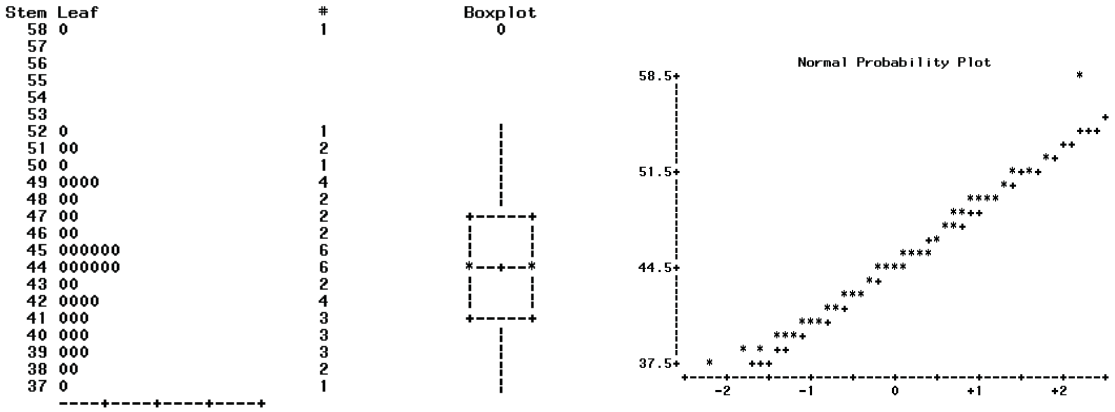
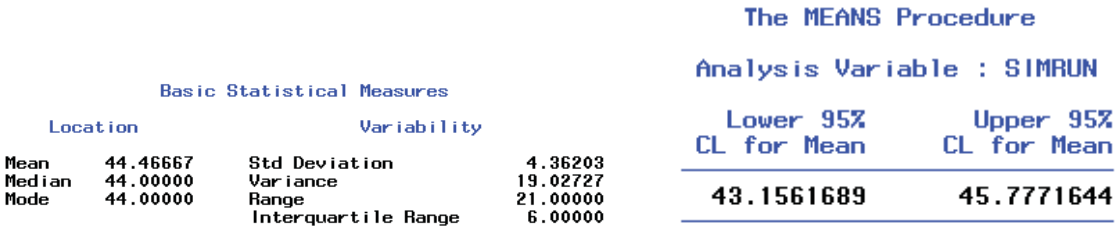


Figure 7.14 Normality Plots for Data

Given that the data is assumed normally distributed, Telly constructed a 95% confidence interval regarding the Maximum number of seats used on a 64 seat bus. SAS indicated a mean of 44.47 with a 95% confidence interval (43.16,45.77) as shown in Figure 7.15.



The MEANS Procedure

Analysis Variable : SIMRUN

Basic Statistical Measures

Location		Variability	
Mean	44.46667	Std Deviation	4.36203
Median	44.00000	Variance	19.02727
Mode	44.00000	Range	21.00000
		Interquartile Range	6.00000

Lower 95% CL for Mean	Upper 95% CL for Mean
43.1561689	45.7771644

Figure 7.15 Confidence Interval for Mean of Maximum Bus Seats

Feeling confident his sample size was large enough, Telly decided to create a baseline statistic representing average waiting time at Bus Stop 1. He selected Bus Stop 1 since this area was visible to customers purchasing tickets and if long lines were visible, it might discourage potential customers. The baseline value was intended to give analysts an idea of the expected average wait time since there is never demand for bus capacity beyond what is possible. In other words, with the current configuration, this is the best the system can perform. This value then can be compared to runs using smaller bus sizes to help determine the best bus to purchase.

Telly established normality for this baseline data set then tabulated an average queue time of 39.89 minutes with a 95% Confidence Interval (39.12, 40.65). Telly wanted to assess the impact of dropping down to 48, 32, and 24 seat buses (those were the available configurations). The simulation was rerun for those scenarios and data was collected. First Telly compared the maximum number of seats used in each scenario. Table 7.8 summarizes the results:

Scenario	Average of Maximum Seats Used
64 Seat Bus	44.47
48 Seat Bus	43.84
32 Seat Bus	32 (Always hit Maximum)
24 Seat Bus	24 (Always hit Maximum)

Table 7.8 Average Maximum Bus Seats Used

Then he used MS-Excel to develop an ANOVA to determine if any significant difference existed between the various seat configurations. Figure 7.16 summarizes:

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
64 Seat Bus	45	1794.96	39.89	6.56
48 Seat Bus	45	1798.233	9.96 7	.36
32 Seat Bus	45	2007.06	44.60	67.18
24 Seat Bus	45	3896.38	86.59	264.91

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	69313	3	23104.31	267.10	2.93E-65	2.66
Within Groups	15224	176	86.50			
Total	84537	179				

Figure 7.16 ANOVA Result for Bus Seat Configurations

The results indicated that on average, customers at Bus Stop #1 waited an average of 39.89 minutes for a 64 seat bus, 39.96 minutes for the 48 seat bus, 44.6 minutes for the 32 seat bus and 86.59 minutes for the 24 seat bus. Telly further analyzed the data and discovered no significant difference existed between the queue times in the 64 and 48 seat buses. Since the 24 seat bus appeared too limited and didn't adequately move the passengers fast enough, Telly wanted to compare the 48 seat and 32 seat buses more closely. Figure 7.17 shows that a significant difference between these two configurations exists and that approximately 5 minutes is saved by using the larger bus.

	48 Seats	**32 Seats**
Mean	39.96	44.60
Variance	7.36	67.18
Observations	45	45
Pooled Variance	37.27	
Hypothesized Mean Difference	0	
Df	88	
t Stat	-3.61	
P(T<=t) one-tail	0.00	
t Critical one-tail	1.66	
P(T<=t) two-tail	0.001	
t Critical two-tail	1.99	

Figure 7.17 48 and 32 Seat Buses Compared

7.5.3 Additional Scenario

Telly developed a report to communicate his findings to Kafy DePorres. In addition to the findings about maximum seats used and average times in queue, other statistics were gathered including average utilization of the bus, average route times, and number of passengers moved.

Kafy spent time reviewing the data and decided that none of the options were really very desirable. The average queue time in the 48 seat bus scenario was still too high at 39.96 minutes. So an additional scenario was devised. Kafy asked Telly to rerun the model with two 24 seat buses that were staggered so multiple locations could be serviced simultaneously. Telly agreed to recode the model so that would work (see code at end in Appendix 7.8). The model was run and compared against the 48 seat version of the single bus model. Figure 7.18 shows the comparison of Bus Stop 1 waiting Times.

	48 Seat	Two 24 Seat Buses
Mean	39.96	29.37
Variance	7.36	66.63
Observations	45	45
Pooled Variance	36.99	
Hypothesized Mean Difference	0	
Df	88	
t Stat	8.26	
P(T<=t) one-tail	6.897E-13	
t Critical one-tail	1.66	
P(T<=t) two-tail	1.379E-12	
t Critical two-tail	1.99	

Figure 7.18 Comparison of Waiting Times Between 2 Bus System and 48 Seat Bus

The new results provided a much better solution – using two 24 seat buses, the average waiting times dropped by over ten minutes. Telly formalized his findings in a report and submitted it to Kafy. Eventually two 24 seat buses were purchased and implemented on the route.

7.6 Operations, Maintenance and Archival Phase

The model was stored, together with documentation and information about the assumptions used. If the routes need to be modified or customer numbers increase, the model can be used to answer additional questions.

7.7 Bibliography

For additional information about the simulation language being used in this chapter, see the following Web resources: http://www.minutemansoftware.com/

7.8 Appendix: GPSS World Source Code Listing (Two Bus Model)

```
**********************************************************************
*       MiniBus Route Model                    *
*       Time units are in Minutes              *
**********************************************************************

        RMULT 9441          ;Reseed Random Number Generators
BusSeat1 STORAGE 24         ;Bus 1 can hold 24 people
BusSeat2 STORAGE 24         ;Bus 2 can hold 24 people
STOP1a STORAGE 1            ;Stop 1a holds 1 bus
STOP2a STORAGE 1            ;Stop 2a holds 1 bus
STOP3a STORAGE 1            ;Stop 3a holds 1 bus
STOP4a STORAGE 1            ;Stop 4a holds 1 bus
STOP5a STORAGE 1            ;Stop 5a holds 1 bus
STOP1b STORAGE 1            ;Stop 1b holds 1 bus
STOP2b STORAGE 1            ;Stop 2b holds 1 bus
STOP3b STORAGE 1            ;Stop 3b holds 1 bus
STOP4b STORAGE 1            ;Stop 4b holds 1 bus
STOP5b STORAGE 1            ;Stop 5b holds 1 bus

* These Boolean Variables check to see if a bus arrives and if room exists on it.
STOP1V BVARIABLE SF$STOP1a&R$BusSeat1'GE'1|SF$STOP1b&R$BusSeat2'GE'1
STOP2V BVARIABLE SF$STOP2a&R$BusSeat1'GE'1|SF$STOP2b&R$BusSeat2'GE'1
STOP3V BVARIABLE SF$STOP3a&R$BusSeat1'GE'1|SF$STOP3b&R$BusSeat2'GE'1
STOP4V BVARIABLE SF$STOP4a&R$BusSeat1'GE'1|SF$STOP4b&R$BusSeat2'GE'1
STOP5V BVARIABLE SF$STOP5a&R$BusSeat1'GE'1|SF$STOP5b&R$BusSeat2'GE'1

* These Boolean Variables are used to see if it is Bus 1 or 2 that has stopped
BUS1S1 BVARIABLE SF$STOP1a&R$BusSeat1'GE'1
BUS1S2 BVARIABLE SF$STOP2a&R$BusSeat1'GE'1
BUS1S3 BVARIABLE SF$STOP3a&R$BusSeat1'GE'1
BUS1S4 BVARIABLE SF$STOP4a&R$BusSeat1'GE'1
BUS1S5 BVARIABLE SF$STOP5a&R$BusSeat1'GE'1

* These Boolean Variables are used to check if a bus is at station a or b
ATSTOP1 BVARIABLE SF$STOP1a|SF$STOP1b
ATSTOP2 BVARIABLE SF$STOP2a|SF$STOP2b
ATSTOP3 BVARIABLE SF$STOP3a|SF$STOP3b
ATSTOP4 BVARIABLE SF$STOP4a|SF$STOP4b
ATSTOP5 BVARIABLE SF$STOP5a|SF$STOP5b
```

```
*************************************************************************
* Bus Route 1 with Starts and Stops and Waits for Passengers
*************************************************************************

            GENERATE    ,,,1                    ;Create Bus Transaction for Station 1
Start
            ASSIGN      BUSNUM,1
Again       ADVANCE     (Exponential(1,12,4))   ; Bus Travels to Stop #1
            ENTER       STOP1a                  ; Bus at Stop
            ADVANCE     (Normal(2,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP1a                  ; Leave Bus Stop

            ADVANCE     (Exponential(3,8,3))    ; Bus Travels to Stop #2
            ENTER       STOP2a                  ; Bus at Stop
            ADVANCE     (Normal(4,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP2a                  ; Leave Bus Stop

            ADVANCE     (Exponential(5,14,4))   ; Bus Travels to Stop #3
            ENTER       STOP3a                  ; Bus at Stop
            ADVANCE     (Normal(6,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP3a                  ; Leave Bus Stop

            ADVANCE     (Exponential(7,8,3))    ; Bus Travels to Stop #4
            ENTER       STOP4a                  ; Bus at Stop
            ADVANCE     (Normal(8,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP4a                  ; Leave Bus Stop

            ADVANCE     (Exponential(9,15,4))   ; Bus Travels to Stop #5
            ENTER       STOP5a                  ; Bus at Stop
            ADVANCE     (Normal(10,2,.3))       ; Passenger Loading/Unloading Time
            LEAVE       STOP5a                  ; Leave Bus Stop
            TRANSFER    ,Again                  ; Move to Travel Back to Stop #1
```

```
************************************************************************
* Bus Route 2 With Starts and Stops and Waits for Passengers
************************************************************************

            GENERATE    ,,,1                    ;Create Bus Transaction for Station 3
Start
            ASSIGN      BUSNUM,2
            TRANSFER    ,Bus2St                 ;Start Day at Station #3

Again2      ADVANCE     (Exponential(1,12,4))   ; Bus Travels to Stop #1
            ENTER       STOP1b                  ; Bus at Stop
            ADVANCE     (Normal(2,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP1b                  ; Leave Bus Stop

            ADVANCE     (Exponential(3,8,3))    ; Bus Travels to Stop #2
            ENTER       STOP2b                  ; Bus at Stop
            ADVANCE     (Normal(4,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP2b                  ; Leave Bus Stop

            ADVANCE     (Exponential(5,14,4))   ; Bus Travels to Stop #3
            ENTER       STOP3b                  ; Bus at Stop
            ADVANCE     (Normal(6,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP3b                  ; Leave Bus Stop

Bus2St      ADVANCE     (Exponential(7,8,3))    ; Bus Travels to Stop #4
            ENTER       STOP4b                  ; Bus at Stop
            ADVANCE     (Normal(8,2,.3))        ; Passenger Loading/Unloading Time
            LEAVE       STOP4b                  ; Leave Bus Stop

            ADVANCE     (Exponential(9,15,4))   ; Bus Travels to Stop #5
            ENTER       STOP5b                  ; Bus at Stop
            ADVANCE     (Normal(10,2,.3))       ; Passenger Loading/Unloading Time
            LEAVE       STOP5b                  ; Leave Bus Stop
            TRANSFER    ,Again2                 ; Move to Travel Back to Stop #1
```

```
*********************************************************************************
        GENERATE 600                                    ; 10 Hour Shift in Minute
        TERMINATE 1                                     ; Stop for Statistics after Each Shift
*********************************************************************************

                GENERATE    (POISSON(11,3.58))          ; Passengers at Stop 1 Arrive
                QUEUE       STOP1Q                      ; Wait for Bus
                TEST E      BV$STOP1V,1                 ; Bus Arrives, If Space Get Onboard
                TEST E      BV$BUS1S1,1,BUS2S1          ; Is this Bus 1?
                ENTER       BusSeat1                    ; If Bus 1, Get in your Seat
                ASSIGN      BUSNUM,1                    ; Passenger is marked with Bus 1
                TRANSFER    ,ONBUS1                     ; Bus 1 Passenger set

BUS2S1          ENTER       BusSeat2                    ; If Bus 2, Get in your Seat
                ASSIGN      BUSNUM,2                    ; Passenger is marked with Bus 2

ONBUS1          DEPART      STOP1Q                      ; No Longer in Queue
                Transfer    .8,Next1a,OutAt2            ; Some go to Stop 2
Next1a          Transfer    .5,Next1b,OutAt3            ; Some go to Stop 3
Next1b          Transfer    .5,OutAt4,OutAt5            ; Some go to Stop 4 or 5

                GENERATE    (POISSON(12,3.22))          ; Passengers at Stop 2
                QUEUE       STOP2Q                      ; Wait for Bus
                TEST E      BV$STOP2V,1                 ; Bus Arrives, If Space Get Onboard
                TEST E      BV$BUS1S2,1,BUS2S2          ; Is this Bus 1?
                ENTER       BusSeat1                    ; If Bus 1, Get in your Seat
                ASSIGN      BUSNUM,1                    ; Passenger is marked with Bus 1
                TRANSFER    ,ONBUS2                     ; Bus 1 Passenger set

BUS2S2          ENTER       BusSeat2                    ; If Bus 2, Get in your Seat
                ASSIGN      BUSNUM,2                    ; Passenger is marked with Bus 2

ONBUS2          DEPART      STOP2Q                      ; No Longer in Queue
                Transfer    .76,Next2a,OutAt3           ; Some go to Stop 3
Next2a          Transfer    .834,Next2b,OutAt4          ; Some go to Stop 4
Next2b          Transfer    .5,OutAt5,OutAt1            ; Some go to Stop 5 or 1
                GENERATE    12,2                        ; Passengers at Stop 3
                QUEUE       STOP3Q                      ; Wait for Bus
                TEST E      BV$STOP3V,1                 ; Bus Arrives, If Space Get Onboard
```

```
            TEST E      BV$BUS1S3,1,BUS2S3    ; Is this Bus 1?
            ENTER       BusSeat1              ; If Bus 1, Get in your Seat
            ASSIGN      BUSNUM,1              ; Passenger is marked with Bus 1
            TRANSFER    ,ONBUS3               ; Bus 1 Passenger set

BUS2S3      ENTER       BusSeat2              ; If Bus 2, Get in your Seat
            ASSIGN      BUSNUM,2              ; Passenger is marked with Bus 2

ONBUS3      DEPART      STOP3Q                ; No Longer in Queue
            Transfer    .67,Next3a,OutAt4     ; Some go to Stop 4
Next3a      Transfer    .818,Next3b,OutAt5    ; Some go to Stop 5
Next3b      Transfer    .67,OutAt1,OutAt2     ; Some go to Stop 1 or 2
            GENERATE    (POISSON(13,3.34))    ; Passengers at Stop 4
            QUEUE       STOP4Q                ; Wait for Bus
            TEST E      BV$STOP4V,1           ; Bus Arrives, If Space Get Onboard
            TEST E      BV$BUS1S4,1,BUS2S4    ; Is this Bus 1?
            ENTER       BusSeat1              ; If Bus 1, Get in your Seat
            ASSIGN      BUSNUM,1              ; Passenger is marked with Bus 1
            TRANSFER    ,ONBUS4               ; Bus 1 Passenger set
```

```
BUS2S4      ENTER       BusSeat2                ; If Bus 2, Get in your Seat
            ASSIGN      BUSNUM,2                ; Passenger is marked with Bus 2

ONBUS4      DEPART      STOP4Q                  ; No Longer in Queue
            Transfer    .76,Next4a,OutAt5       ; Some go to Stop 5
Next4a      Transfer    .625,Next4b,OutAt1      ; Some go to Stop 1
Next4b      Transfer    .67,OutAt2,OutAt3       ; Some go to Stop 2 or 3
            GENERATE    (POISSON(14,3.12))      ; Passengers at Stop 5
            QUEUE       STOP5Q                  ; Wait for Bus
            TEST E      BV$STOP5V,1             ; Bus Arrives, If Space Get Onboard
            TEST E      BV$BUS1S5,1,BUS2S5      ; Is this Bus 1?
            ENTER       BusSeat1                ; If Bus 1, Get in your Seat
            ASSIGN      BUSNUM,1                ; Passenger is marked with Bus 1
            TRANSFER    ,ONBUS5                 ; Bus 1 Passenger set

BUS2S5      ENTER       BusSeat2                ; If Bus 2, Get in your Seat
            ASSIGN      BUSNUM,2                ; Passenger is marked with Bus 2

ONBUS5      DEPART      STOP5Q                  ; No Longer in Queue
            Transfer    .88,Next5a,OutAt1       ; Some go to Stop 1
Next5a      Transfer    .67,Next5b,OutAt2       ; Some go to Stop 2
Next5b      Transfer    .5,OutAt3,OutAt4        ; Some go to Stop 3 or 4

**********************************************************************

OutAt1      QUEUE       WAIT41                  ; Tracking Travel Time
            TEST E      BV$ATSTOP1,1            ; Passenger Waits for Stop
            TEST E      P$BUSNUM,1,OBus2A1      ; On Bus 1?
            LEAVE       Busseat1                ; Gets out of Seat on Bus 1
            TRANSFER    ,OutAt1b                ; Leave Bus 1

OBus2A1     LEAVE       Busseat2                ; Gets out of Seat on Bus 2
OutAt1b     DEPART      WAIT41                  ; Leaves Bus Area
            TERMINATE                           ; Is No Longer in Model
```

```
OutAt2      QUEUE       WAIT42              ; Tracking Travel Time
            TEST E      BV$ATSTOP2,1        ; Passenger Waits for Stop
            TEST E      P$BUSNUM,1,OBus2A2  ; On Bus 1?
            LEAVE       Busseat1            ; Gets out of Seat on Bus 1
            TRANSFER    ,OutAt2b            ; Leave Bus 1

OBus2A2     LEAVE       Busseat2            ; Gets out of Seat on Bus 2
OutAt2b     DEPART      WAIT42              ; Leaves Bus Area
            TERMINATE                       ; Is No Longer in Model

OutAt3      QUEUE       WAIT43              ; Tracking Travel Time
            TEST E      BV$ATSTOP3,1        ; Passenger Waits for Stop
            TEST E      P$BUSNUM,1,OBus2A3  ; On Bus 1?
            LEAVE       Busseat1            ; Gets out of Seat on Bus 1
            TRANSFER    ,OutAt3b            ; Leave Bus 1

OBus2A3     LEAVE       Busseat2            ; Gets out of Seat on Bus 2
OutAt3b     DEPART      WAIT43              ; Leaves Bus Area
            TERMINATE                       ; Is No Longer in Model

OutAt4      QUEUE       WAIT44              ; Tracking Travel Time
            TEST E      BV$ATSTOP4,1        ; Passenger Waits for Stop
            TEST E      P$BUSNUM,1,OBus2A4  ; On Bus 1?
            LEAVE       Busseat1            ; Gets out of Seat on Bus 1
            TRANSFER    ,OutAt4b            ; Leave Bus 1

OBus2A4     LEAVE       Busseat2            ; Gets out of Seat on Bus 2
OutAt4b     DEPART      WAIT44              ; Leaves Bus Area
            TERMINATE                       ; Is No Longer in Model

OutAt5      QUEUE       WAIT45              ; Tracking Travel Time
            TEST E      BV$ATSTOP2,1        ; Passenger Waits for Stop
            TEST E      P$BUSNUM,1,OBus2A5  ; On Bus 1?
            LEAVE       Busseat1            ; Gets out of Seat on Bus 1
            TRANSFER    ,OutAt5b            ; Leave Bus 1

OBus2A5     LEAVE       Busseat2            ; Gets out of Seat on Bus 2
OutAt5b     DEPART      WAIT45              ; Leaves Bus Area
            TERMINATE                       ; Is No Longer in Model
```

Acknowledgements

I would like to thank several people for their contributions to this textbook. Amy Rucker, a graduate assistant at K-State, provided help developing illustrations used in the text. My wife, Annette McHaney, proofread the manuscript and provided a valuable critique, provided inspiration, and suggested helpful changes.